

Bộ Giáo Dục Và Đào Tạo  
Trường Đại Học Ngoại Ngữ - Tin Học Thành Phố Hồ Chí Minh  
Khoa Công Nghệ Thông Tin



**MÔN HỌC: PENETRATION TESTING**  
**ĐỀ TÀI: THỰC HIỆN SCAN SOURCE CODE VỚI CHECK MARX**

**Giáo Viên Hướng Dẫn:** ThS. Phạm Đình Thắng

**Thành Viên:**

- Nguyễn Thị Kim Doanh – MSSV: 22DH110511
- Nguyễn Thúy Vy – MSSV: 22DH114363

Tp. Hồ Chí Minh, Ngày 01 Tháng 04 Năm 2025

## LỜI CẢM ƠN

Chúng em xin gửi lời cảm ơn chân thành và sâu sắc nhất đến tất cả thầy cô Trường Đại Học Ngoại Ngữ - Tin Học TP. Hồ Chí Minh nói chung cùng thầy cô trong Khoa Công Nghệ Thông Tin nói riêng đã tận tình giảng dạy, truyền đạt những kiến thức và kinh nghiệm quý báu cho chúng em trong suốt quá trình học tập tại trường.

Trong suốt thời gian nhóm làm bài báo cáo đề tài môn, chúng em xin gửi lời cảm ơn chân thành và sâu sắc đến Thầy Phạm Đình Thắng, người đã hết lòng giúp đỡ và theo sát nhóm chúng em trong suốt quá trình thực hiện đề tài đồ án môn học này, chỉ ra cho nhóm hướng đi để nhóm có thể hoàn thành tốt nhất bài báo cáo đề tài đồ án này đúng thời hạn quy định.

Trong quá trình thực hiện đề tài môn Penetration Testing, dù nhóm đã cố gắng hoàn thiện đề tài một cách tối ưu nhất nhưng do thời gian và kiến thức còn hạn chế nên sẽ không tránh khỏi những thiếu sót và sai sót nhất định, rất mong nhận được sự cảm thông từ những đóng góp ý chân thành từ quý thầy cô khoa Công Nghệ Thông Tin.

Sau cùng em xin gửi lời cảm ơn đến tất cả các bạn đã tham gia đóng góp ý kiến và giúp đỡ em trong suốt quá trình thực hiện đề tài môn Penetration Testing.

Em xin chân thành cảm ơn tất cả mọi người

## MỤC LỤC

Thành Viên: .....	1
LỜI CẢM ƠN.....	2
<b>MỤC LỤC</b> .....	<b>3</b>
DANH MỤC HÌNH ẢNH.....	4
CHƯƠNG I: GIỚI THIỆU VỀ ĐỀ TÀI.....	7
1. Giới thiệu về kiểm thử xâm nhập: .....	7
2. Mục tiêu của kiểm thử xâm nhập: .....	7
3. Lợi ích của kiểm thử xâm nhập: .....	7
CHƯƠNG II: CƠ SỞ LÝ THUYẾT .....	9
1. Các hình thức kiểm thử thâm nhập: .....	9
2. Ưu, nhược điểm các hình thức kiểm thử xâm nhập: .....	10
3. Giai đoạn thực hiện:.....	10
3.1. Thu thập thông tin (Reconnaissance):.....	11
3.2. Phân tích lỗ hổng (Vulnerability Analysis): .....	11
3.3. Tân công (Exploitation):.....	11
3.4. Đánh giá mức độ tổn thương (Post-Exploitation): .....	11
3.5. Báo cáo và đề xuất (Reporting): .....	12
4. Công cụ .....	12
CHƯƠNG III: TRIỂN KHAI.....	14
1. Cài đặt các công cụ .....	14
1.1. Cài đặt Checkmarx.....	14
1.2. Cài đặt Checkmarx .....	21
2. Triển khai đề tài với checkmarx .....	29
2.1. Tạo Project với checkmarx .....	29
2.2. Phân tích lỗi .....	32
2.3. Phân tích từng lỗi có trên Project.....	36
2.3.1. JavaScript .....	36
2.3.2. PHP .....	46
CHƯƠNG IV: ĐÁNH GIÁ.....	60
CHƯƠNG V: KẾT LUẬN.....	61
TÀI LIỆU THAM KHẢO .....	62

## DANH MỤC HÌNH ẢNH

Hình 1. Giới thiệu Penetration Testing .....	7
Hình 2. Các hình thức kiểm thử thâm nhập .....	10
Hình 3. Quy trình kiểm thử thâm nhập.....	12
Hình 4. Một số công cụ kiểm thử xâm nhập.....	13
Hình 5. Yêu cầu cấu hình .....	14
Hình 6. Cài đặt .Net .....	14
Hình 7. Cài đặt Microsoft Visual C++ .....	14
Hình 8. Cài đặt Microsoft Visual C++ thành công .....	15
Hình 9. Cài IIS trên Windos Server.....	15
Hình 10. Cài IIS trên Windos Server thành công .....	16
Hình 11. Kết nối đến SQL Server.....	16
Hình 12. Tùy chọn Properties .....	17
Hình 13.....	18
Hình 14. Hoàn tất.....	18
Hình 15. Restart SQL Server .....	18
Hình 16. Tùy chọn Properties .....	19
Hình 17. Thiết lập mật khẩu.....	19
Hình 18. Chọn Enable .....	20
Hình 19. User sa đã mất tích đỏ .....	20
Hình 20. Màn hình đăng nhập SQL Server.....	21
Hình 21. Cài đặt Checkmarx.....	21
Hình 22. Cài đặt Checkmarx.....	22
Hình 23. Cài đặt Checkmarx.....	22
Hình 24. Cài đặt Checkmarx.....	23
Hình 25. Cài đặt Checkmarx.....	23
Hình 26. Cài đặt Checkmarx.....	24
Hình 27. Cài đặt Checkmarx.....	24
Hình 28. Cài đặt Checkmarx.....	25
Hình 29. Cài đặt Checkmarx.....	25
Hình 30. Cài đặt Checkmarx.....	26
Hình 31. Cài đặt Checkmarx.....	26
Hình 32. Cài đặt Checkmarx.....	27
Hình 33. Cài đặt Checkmarx thành công .....	27
Hình 34. Chạy folder crack .....	28
Hình 35. Màn hình đăng nhập Checkmarx .....	28
Hình 36. Tạo Project .....	29
Hình 37. Đặt tên Project .....	29
Hình 38. Upload file cần scan lỗi.....	30
Hình 39. Màn hình hiển thị đang trong quá trình scan lỗi .....	30
Hình 40. Màn hình scan lỗi thành công.....	31
Hình 41. Xem kết quả các file đã scan tại đây. ....	31
Hình 42. Danh sách các file đã được scan thành công .....	32
Hình 43. Thông tin Project.....	32
Hình 44. Nhấn tùy chọn Open Viewer .....	34
Hình 45. Màn hình hiển thị sau khi ấn Open Viewer .....	34
Hình 46. Màn hình hiển thị sau khi nhấn Graph.....	35

Hình 47. Xem chi tiết và cách fix lỗi tại đây.....	35
Hình 48. Client_Hardcoded_Domain: Lỗi hỏng bảo mật từ Checkmarx Enterprise.....	36
Hình 49. Lệnh import từ xa.....	37
Hình 50. Lệnh import cục bộ .....	37
Hình 51. Client_JQuery_Deprecated_Symbols: Các vấn đề bảo mật liên quan đến việc sử dụng các biểu tượng/hàm jQuery đã bị loại bỏ. ....	38
Hình 52. Client_JQuery_Deprecated_Symbols: Các vấn đề bảo mật liên quan đến việc sử dụng các biểu tượng/hàm jQuery đã bị loại .....	38
Hình 53. Client_JQuery_Deprecated_Symbols: Các vấn đề bảo mật liên quan đến việc sử dụng các biểu tượng/hàm jQuery đã bị loại .....	39
Hình 54. jQuery - Sử dụng \$.parseJSON bị loại bỏ .....	40
Hình 55. Sử dụng Native Call thay vì jQuery Calls bị loại bỏ .....	40
Hình 56. Lấy năm qua phương thức JavaScript bị loại bỏ .....	40
Hình 57. Lấy năm qua phương thức JavaScript được hỗ trợ .....	40
Hình 58. Gọi một hàm bị loại bỏ, được chú thích bằng JSDoc.....	41
Hình 59. Potential_Clickjacking_on_Legacy_Browsers: Lỗi hỏng Clickjacking trên các trình duyệt cũ (Legacy Browsers).....	41
Hình 60. Potential_Clickjacking_on_Legacy_Browsers: Lỗi hỏng Clickjacking trên các trình duyệt cũ (Legacy Browsers).....	42
Hình 61. Potential_Clickjacking_on_Legacy_Browsers: Lỗi hỏng Clickjacking trên các trình duyệt cũ (Legacy Browsers).....	42
Hình 62. Potential_Clickjacking_on_Legacy_Browsers: Lỗi hỏng Clickjacking trên các trình duyệt cũ (Legacy Browsers).....	43
Hình 63. Web dễ bị Clickjacking .....	44
Hình 64. Framebuster cơ bản.....	44
Hình 65. Framebuster bảo mật tốt hơn .....	45
Hình 66. File_Manipulation: Lỗi hỏng File Manipulation có thể dẫn đến ghi đè hoặc thực thi mã độc nếu không được kiểm soát chặt chẽ .....	46
Hình 67. File_Manipulation: Lỗi hỏng File Manipulation có thể dẫn đến ghi đè hoặc thực thi mã độc nếu không được kiểm soát chặt chẽ .....	46
Hình 68. Sử dụng basename() để ngăn tấn công Directory Traversal .....	47
Hình 69. Sử dụng danh sách trắng (whitelist) .....	47
Hình 70. Mã dễ bị tấn công .....	48
Hình 71. Mã an toàn hơn.....	48
Hình 72. Missing_HSTS_Header: Lỗi hỏng Missing HSTS Header .....	48
Hình 73. Missing_HSTS_Header: Lỗi hỏng Missing HSTS Header .....	49
Hình 74. Missing_HSTS_Header: Lỗi hỏng Missing HSTS Header .....	49
Hình 75. Thiết lập HSTS trong PHP .....	50
Hình 76. Path_Traversal: Lỗi hỏng Path Traversal.....	51
Hình 77. Path_Traversal: Lỗi hỏng Path Traversal.....	51
Hình 78. Path_Traversal: Lỗi hỏng Path Traversal.....	52
Hình 79. Path_Traversal: Lỗi hỏng Path Traversal.....	52
Hình 80. Tấn công Path Traversal với đường dẫn tuyệt đối trong tham số "filename" .....	54
Hình 81. Tấn công Path Traversal với đường dẫn tương đối trong tham số "filename" .....	55
Hình 82. - Tấn công Path Traversal với đường dẫn tuyệt đối trong tham số "filename", có thể dẫn đến tấn công SSRF .....	55
Hình 83. Giảm thiểu tấn công Path Traversal bằng cách sử dụng basename.....	55
Hình 84. Possible_Flow_Control: Kiểm Soát Luồng Chương Trình BịẢnhHưởng .....	56

Hình 85. Possible_Flow_Control: Kiểm Soát Luồng Chương Trình Bị Ánh Hưởng .....	57
Hình 86. Mã dễ bị tấn công (User input ảnh hưởng vòng lặp while) .....	58
Hình 87. Cách khắc phục (Luồng điều khiển tách biệt với user input) .....	58

## CHƯƠNG I: GIỚI THIỆU VỀ ĐỀ TÀI

### 1. Giới thiệu về kiểm thử xâm nhập:

Kiểm tra thâm nhập còn gọi là kiểm thử thâm nhập (Penetration testing, pen test hay ethical hacking) là quá trình thực thi mô phỏng tấn công an ninh mạng vào một hệ thống máy tính hoặc phần mềm định trước nhằm mục đích kiểm tra, đánh giá mức độ an toàn của hệ thống cũng như tìm các lỗ hổng bảo mật (nếu có).



Hình 1. Giới thiệu Penetration Testing

### 2. Mục tiêu của kiểm thử xâm nhập:

Mục đích của pentest không chỉ là kiểm tra các lỗ hổng trong môi trường mà còn để kiểm tra con người và quy trình trước các mối đe dọa có thể xảy ra đối với tổ chức. Việc biết đối thủ nào có nhiều khả năng nhắm mục tiêu vào bạn sẽ cho phép người kiểm thử thâm nhập bắt chước các chiến thuật, kỹ thuật và quy trình (TTP) cụ thể của những đối thủ cụ thể đó – giúp tổ chức có ý tưởng thực tế hơn nhiều về cách vi phạm có thể xảy ra.

### 3. Lợi ích của kiểm thử xâm nhập:

Khi thực hiện pentest định kỳ và đúng cách, doanh nghiệp có thể đạt được những mục tiêu bảo mật quan trọng:

- Tăng cường an ninh cho ứng dụng web, mobile, network, IoT, API, hệ thống cloud, SaaS, phần cứng, v.v. Giảm thiểu tối đa khả năng bị hacker xâm nhập trái phép và gây thiệt hại.
- Các nhà lãnh đạo có cái nhìn toàn cảnh về an ninh ứng dụng & sản phẩm công nghệ của tổ chức.

- Uớc tính thiệt hại mà một cuộc tấn công thực tế có thể gây ra.
- Bảo mật cơ sở dữ liệu, các thông tin quan trọng của doanh nghiệp và thông tin người dùng.
- Giúp hệ thống hoạt động ổn định, giảm thiểu khả năng bị tấn công gây gián đoạn.
- Tìm được các lỗ hổng nguy hiểm mà công cụ/phần mềm phòng thủ tự động khó phát hiện ra.
- Đảm bảo tiêu chuẩn bảo mật của từng ngành cụ thể (PCI DSS, HIPAA, ISO 27001,...).
- Cung cấp niềm tin cho khách hàng, đối tác, nhà đầu tư.

## CHƯƠNG II: CƠ SỞ LÝ THUYẾT

### 1. Các hình thức kiểm thử thâm nhập:

- Black box (kiểm thử hộp đen): đây là phương thức kiểm tra "đóng" nhất. Ở loại kiểm thử này, người thử nghiệm không được cung cấp trước bất kỳ thông tin nào. Trong trường hợp người kiểm thử sẽ tiếp cận thâm nhập như một kẻ tấn công không có đặc quyền, truy cập từ ngoài hệ thống và thực thi tấn công cho đến khai thác được lỗ hổng bảo mật. Kịch bản dạng này có thể được coi là xác thực nhất, chứng minh kẻ tấn công không có kiến thức bên trong vẫn có thể thâm nhập được vào hệ thống.
- White box (kiểm thử hộp trắng): đây là phương thức kiểm tra "mở" nhất. Ở kiểm thử thâm nhập hộp trắng, người thử nghiệm sẽ được cung cấp trước các thông tin về hệ thống, bao gồm việc chia sẻ thông tin mạng và hệ thống đầy đủ như bản đồ mạng, thông tin đăng nhập, thậm chí cả mã nguồn hệ thống. Điều này giúp tiết kiệm thời gian và giảm chi phí tổng thể cho các bước lập kế hoạch và trinh sát. Thủ nghiệm thâm nhập hộp trắng rất hữu ích để mô phỏng một cuộc tấn công có chủ đích vào một hệ thống cụ thể bằng cách sử dụng càng nhiều phương thức tấn công càng tốt.
- Gray box (kiểm thử hộp xám): đây là phương thức tấn công kết hợp, lai giữa hộp đen và hộp trắng. Ở loại kiểm thử này người thử nghiệm được cung cấp trước một số thông tin về hệ thống nhưng không đầy đủ. Thông thường, người thử nghiệm sẽ được cung cấp các thông tin giả định như là một tin tặc (hacker) được cung cấp tài khoản một người dùng thông thường và tiến hành tấn công vào hệ thống dưới vai trò như một nhân viên của doanh nghiệp.

Ngoài ra còn các hình thức pentest khác như: double-blind testing, external testing, internal testing, targeted testing tuy nhiên chúng không phổ biến tại Việt Nam và chỉ được sử dụng với nhu cầu đặc thù của một số doanh nghiệp.



*Hình 2. Các hình thức kiểm thử thâm nhập.*

## **2. Ưu, nhược điểm các hình thức kiểm thử xâm nhập:**

### **Black Box Testing**

- **Ưu điểm:** Không cần biết mã nguồn, tập trung vào hành vi hệ thống, dễ dàng thực hiện.
- **Nhược điểm:** Không phát hiện lỗi về cấu trúc, khó kiểm tra tất cả tình huống, hạn chế với hiệu suất và bảo mật.

### **White Box Testing**

- **Ưu điểm:** Phát hiện lỗi trong mã nguồn, kiểm tra chi tiết mọi phần của hệ thống, cải thiện chất lượng mã.
- **Nhược điểm:** Cần kiến thức lập trình, tốn thời gian với hệ thống phức tạp, bỏ qua hành vi người dùng.

### **Gray Box Testing**

- **Ưu điểm:** Kết hợp ưu điểm của cả hai phương pháp, phát hiện lỗi hiệu quả hơn, phù hợp cho hệ thống phức tạp.
- **Nhược điểm:** Khó xác định phạm vi kiểm thử, cần kỹ năng cao, không phải lúc nào cũng phát hiện lỗi bảo mật.

## **3. Giai đoạn thực hiện:**

Quy trình thực hiện kiểm tra đánh giá an toàn bảo mật thông tin bằng Kiểm tra thâm nhập thông thường có 5 bước chính:

### **3.1. Thu thập thông tin (Reconnaissance):**

Bước đầu tiên trong quá trình pentest là thu thập thông tin về hệ thống mục tiêu. Mục đích của bước này là cung cấp cho người kiểm tra một cái nhìn tổng quan về hệ thống và xác định các điểm yếu có thể được khai thác.

Các công cụ thông thường được sử dụng trong bước này bao gồm:

- WHOIS: tìm thông tin chủ sở hữu miền và các thông tin liên quan.
- Nmap: quét cổng và tìm ra các dịch vụ đang chạy trên hệ thống.
- Recon-ng: một công cụ thu thập thông tin tổng hợp và có tính năng mở rộng.

### **3.2. Phân tích lỗ hổng (Vulnerability Analysis):**

Sau khi thu thập thông tin, người kiểm tra sẽ sử dụng các công cụ để phân tích lỗ hổng trên hệ thống.

Các công cụ thông thường được sử dụng trong bước này bao gồm:

- Nessus: một công cụ quét lỗ hổng tổng hợp.
- OpenVAS: một công cụ quét lỗ hổng tổng hợp, miễn phí và mã nguồn mở.
- Nikto: một công cụ quét lỗ hổng trên các ứng dụng web.

### **3.3. Tấn công (Exploitation):**

Sau khi đã phân tích các lỗ hổng, người kiểm tra sẽ tiến hành khai thác các lỗ hổng đó để xác minh tính khả dụng và khai thác được của chúng. Mục đích của bước này là xác định các cách thức để tấn công hệ thống và xác minh mức độ nguy hiểm của các lỗ hổng bảo mật đó.

Các công cụ thông thường được sử dụng trong bước này bao gồm:

- Metasploit: một nền tảng khai thác lỗ hổng bảo mật.
- Exploit-DB: một cơ sở dữ liệu lỗ hổng bảo mật và các công cụ khai thác liên quan.
- SET (Social-Engineer Toolkit): một công cụ tấn công dựa trên kỹ thuật xâm nhập xã hội.

### **3.4. Đánh giá mức độ tổn thương (Post-Exploitation):**

Sau khi tấn công thành công hệ thống, người kiểm tra sẽ đánh giá mức độ tổn thương của hệ thống và xác định các tài liệu hoặc thông tin quan trọng mà kẻ tấn công có thể truy cập được.

Các công cụ thông thường được sử dụng trong bước này bao gồm:

- Meterpreter: một công cụ tấn công cơ bản và quản lý từ xa được tích hợp trong Metasploit.
- Empire: một nền tảng tấn công mạnh mẽ và linh hoạt cho các cuộc tấn công bằng phần mềm độc hại.
- PowerSploit: một tập hợp các tác vụ và script tấn công dựa trên PowerShell.

### 3.5. Báo cáo và đề xuất (Reporting):

Cuối cùng, người kiểm tra sẽ tạo báo cáo về các lỗ hổng và giải pháp khắc phục. Báo cáo nên cung cấp các thông tin chi tiết về các lỗ hổng bảo mật, bao gồm các cách để khắc phục chúng và các giải pháp để tăng cường bảo mật hệ thống. Nó cũng cần cung cấp các chứng minh và bằng chứng về các hoạt động tấn công đã được thực hiện trong quá trình penetration testing



Hình 3. Quy trình kiểm thử thâm nhập.

## 4. Công cụ

Có rất nhiều công cụ đánh giá bảo mật có sẵn để hỗ trợ kiểm tra thâm nhập, bao gồm cả các phần mềm miễn phí và phần mềm thương mại. Một số nền tảng phần mềm phổ biến phải kể đến như:

- Nmap: giúp scan lỗ hổng bảo mật, scan port và xác định OS.
- Nessus: dựa trên internet và mạng truyền thông để xử lý các lỗ hổng trong vấn đề bảo mật thông tin.
- Pass the Hash: có chức năng hỗ trợ tester bẻ khóa mật khẩu, kiểm tra độ chắc chắn của các lớp bảo mật.
- Metasploit: cung cấp một loạt các module và payloads để thực hiện nhiều loại tấn công
- Wireshark: theo dõi và phân tích gói tin để phát hiện các mối đe dọa và tấn công.
- Burp Suite: gồm các chức năng như kiểm thử bảo mật, kiểm thử tấn công, và kiểm thử tự động.
- OWASP ZAP (Zed Attack Proxy): cung cấp các tính năng như kiểm thử bảo mật, phân tích gói tin, và kiểm thử tự động.

Ngoài ra một số bản phân phối hệ điều hành hướng tới thử nghiệm thâm nhập có sẵn nhiều công cụ đặc biệt như:

- Kali Linux (được thay thế bằng BackTrack vào tháng 12 năm 2012) phát triển dựa trên Debian.
- Parrot Security OS phát triển dựa trên Debian.
- Pentoo phát triển dựa trên Gentoo.



Hình 4. Một số công cụ kiểm thử xâm nhập.

## CHƯƠNG III: TRIỂN KHAI

### 1. Cài đặt các công cụ

#### 1.1. Cài đặt Checkmarx

##### 1.1.1. Yêu cầu cấu hình trên Windows Server

Device	Summary
Memory	8 GB
Processors	4
Hard Disk (NVMe)	100 GB

Hình 5. Yêu cầu cấu hình

##### 1.1.2. Cài đặt các phần mềm hỗ trợ

- Cài đặt .Net

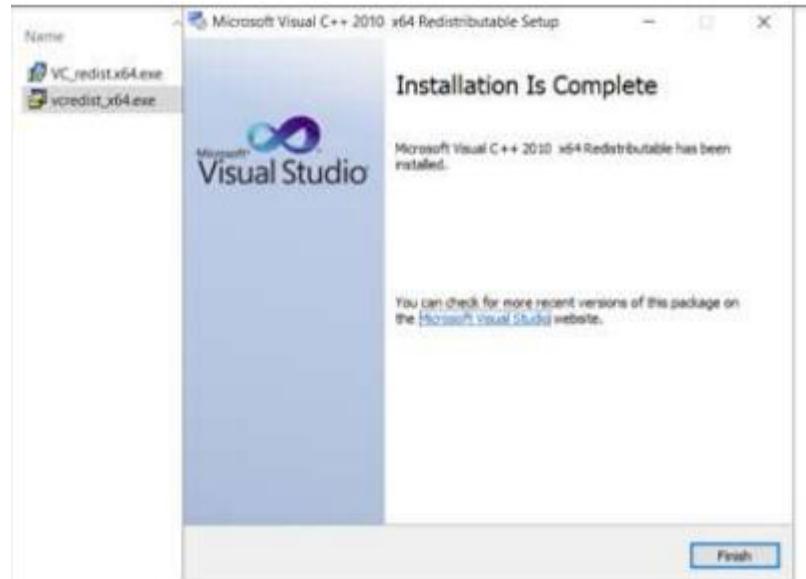


Hình 6. Cài đặt .Net

- Cài đặt Microsoft Visual C++

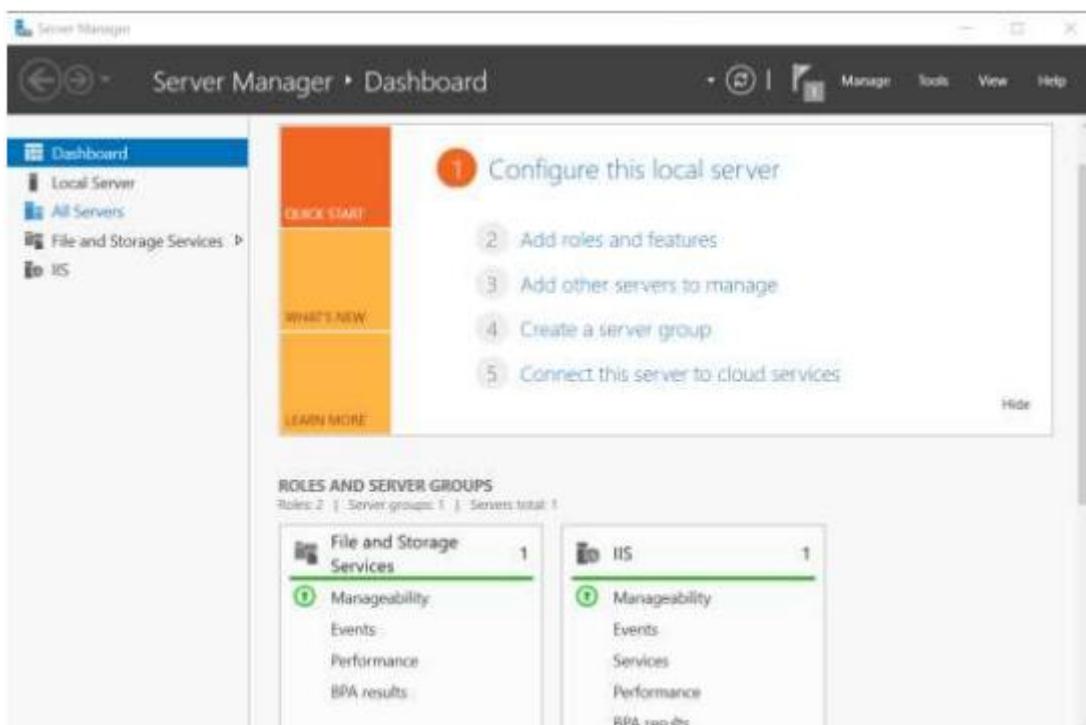


Hình 7. Cài đặt Microsoft Visual C++

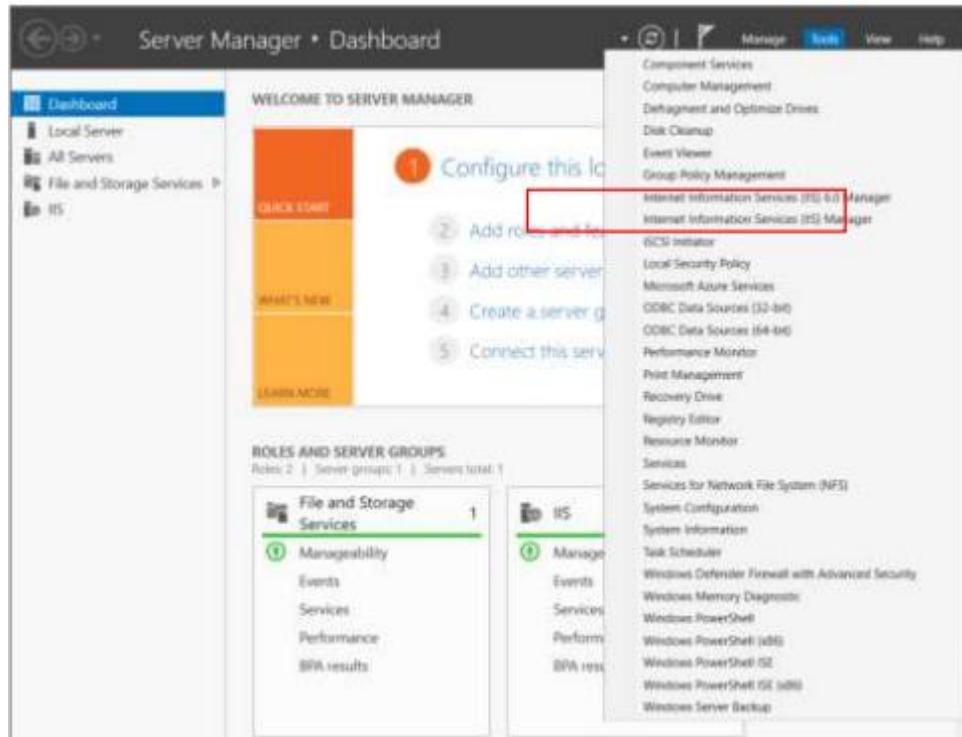


Hình 8. Cài đặt Microsoft Visual C++ thành công

### 1.1.3. Cài IIS trên Windos Server



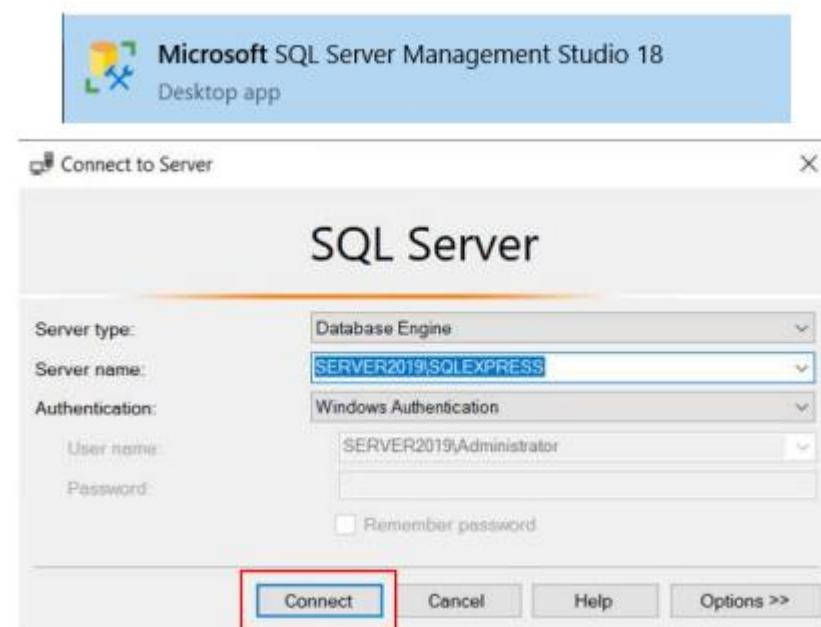
Hình 9. Cài IIS trên Windos Server



Hình 10. Cài IIS trên Windos Server thành công

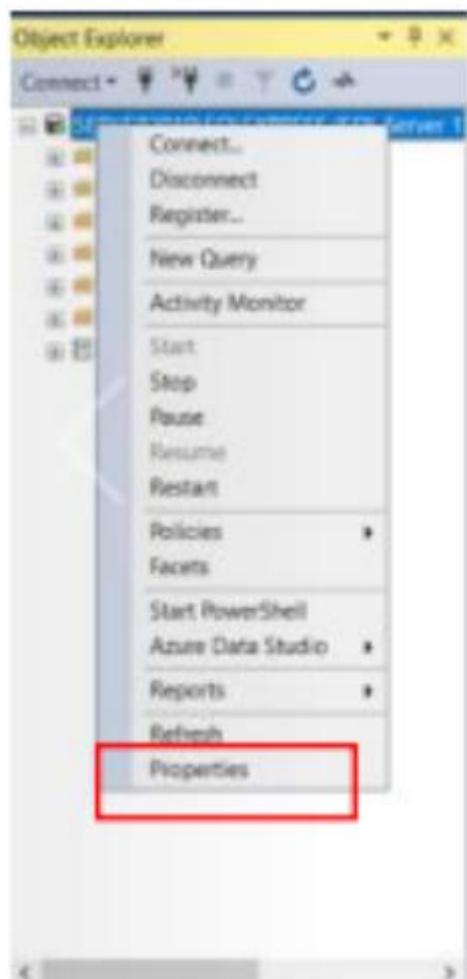
#### 1.1.4. Cài đặt SQL Server

- Sau khi cài đặt xong, tìm Microsoft SQL Server Management Studio mở nó lên và connect vào SQL



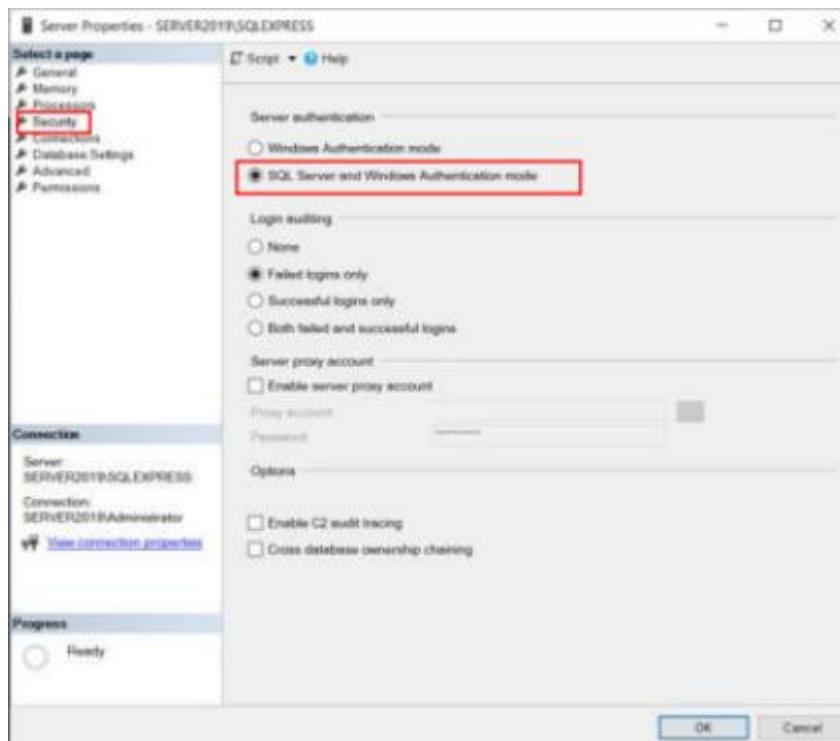
Hình 11. Kết nối đến SQL Server

- Thiết lập sử dụng tài khoản quản trị sa
  - Nhấn chuột phải và chọn Properties



Hình 12. Tùy chọn Properties

- Chọn Security, mục Server authentication, chọn SQL Server...



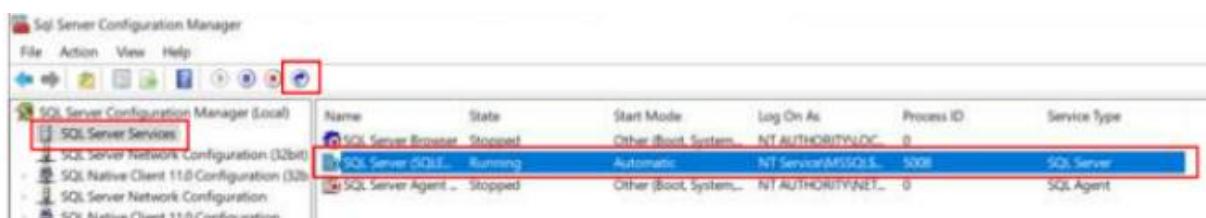
Hình 13

- Nhấn ok



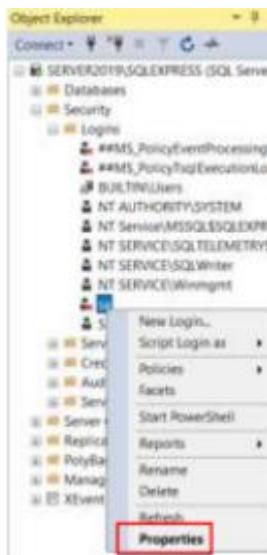
Hình 14. Hoàn tất

- Search SQL Server 2019 Configuration Manager, chọn SQL Server và restart nó



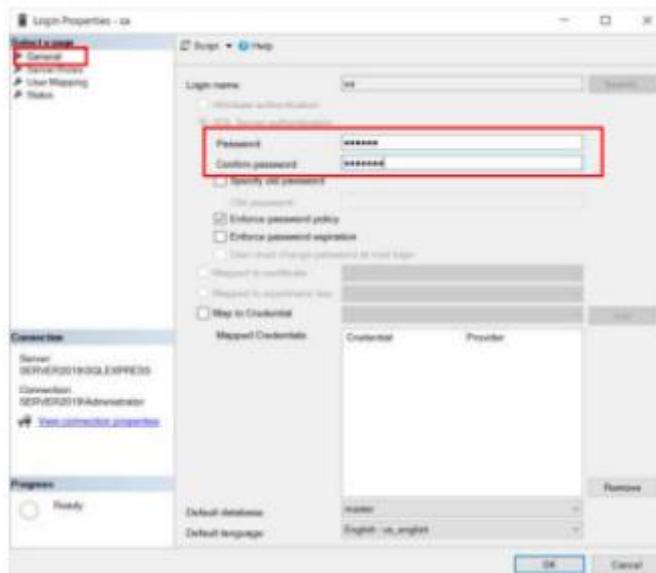
Hình 15. Restart SQL Server

- Quay về SQL Server, chọn Security -> Logins -> sa -> properties



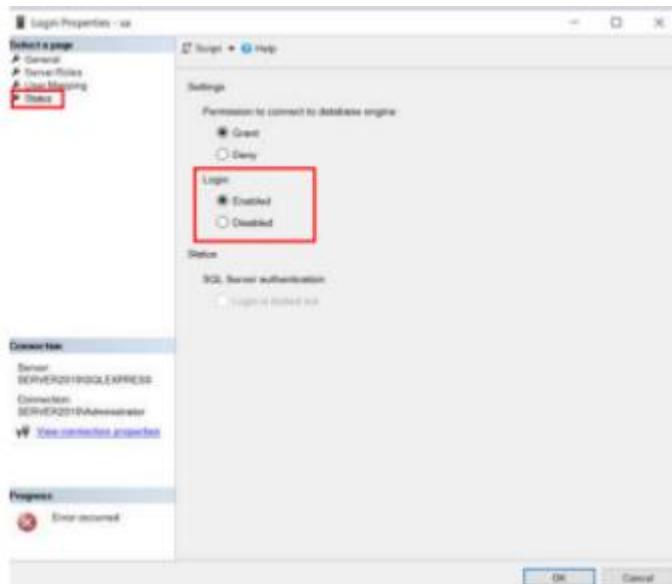
Hình 16. Tùy chọn Properties

- Chọn General và tiến hành set mật khẩu



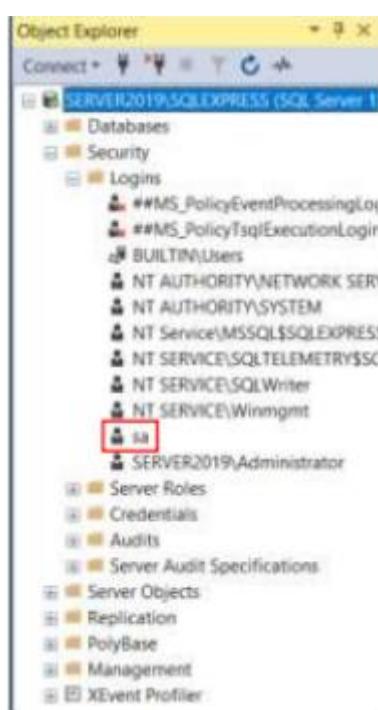
Hình 17. Thiết lập mật khẩu

- Chọn Status -> Enable



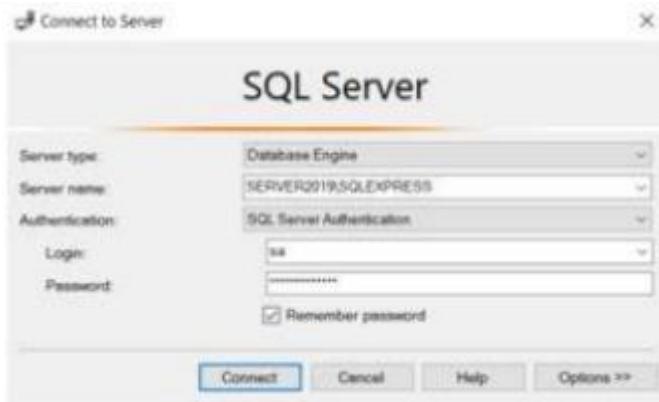
Hình 18. Chọn Enable

- Reset lại trang, user sa sẽ bỏ tích đở



Hình 19. User sa đã mất tích đở

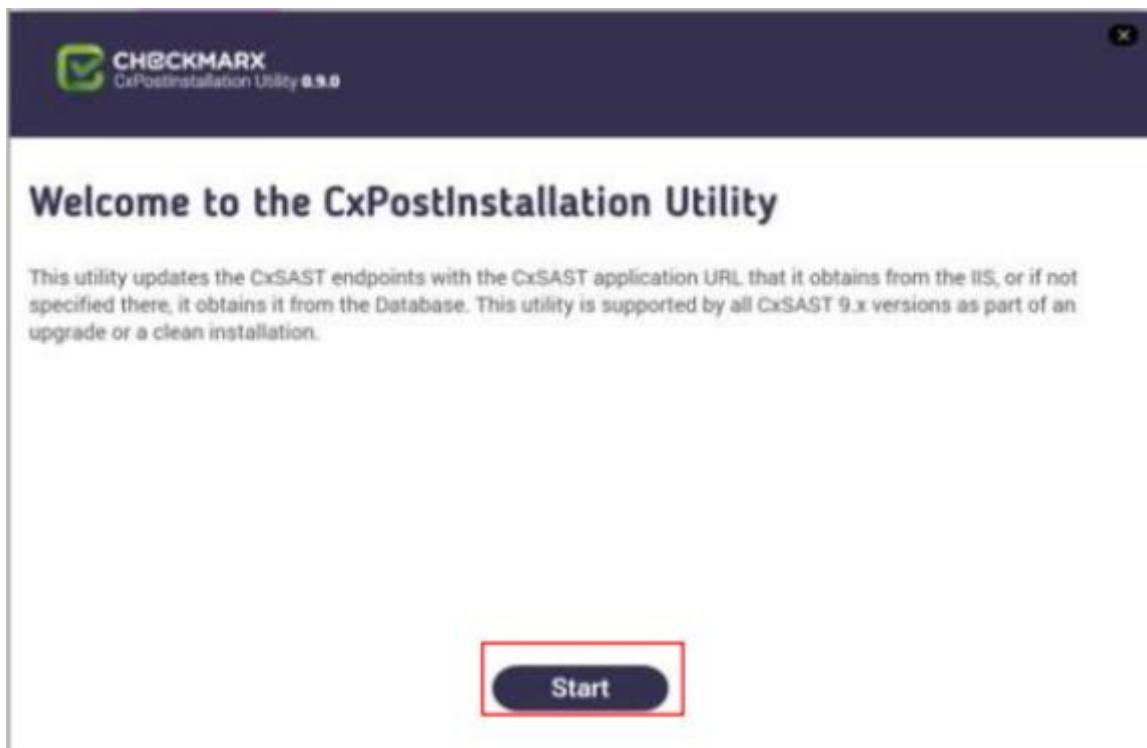
- Ở mục Authentication chọn SQL Server Authentication, đăng nhập



Hình 20. Màn hình đăng nhập SQL Server

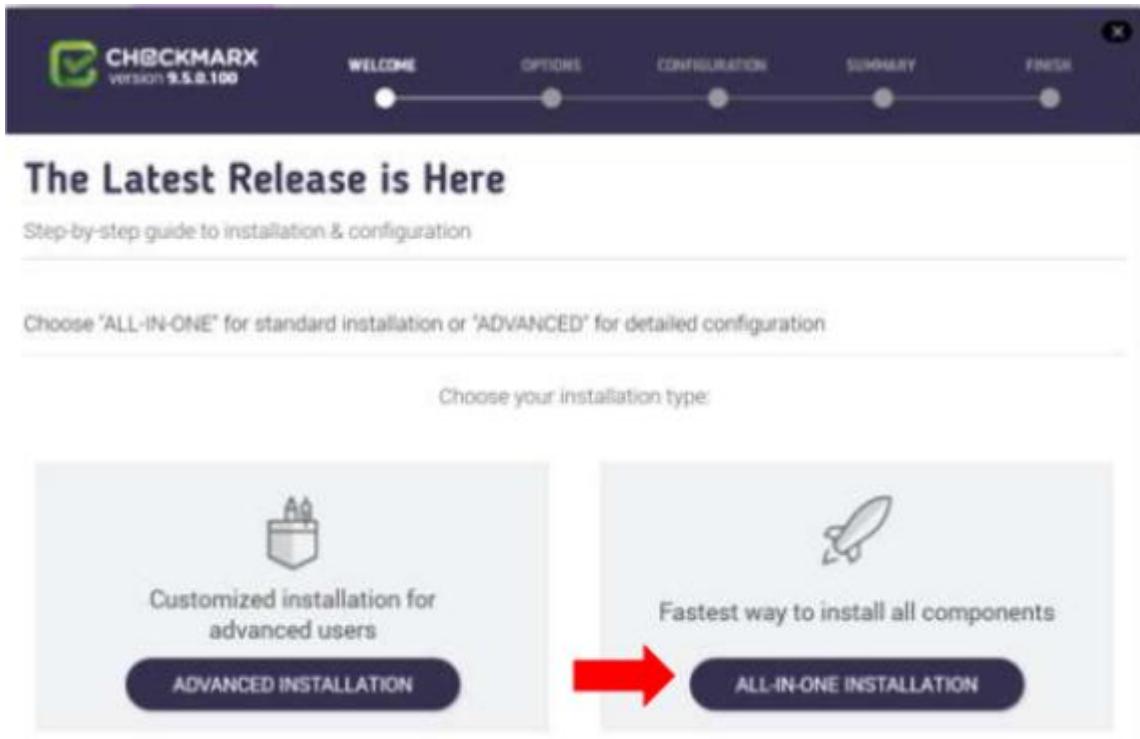
## 1.2. Cài đặt Checkmarx

- Chọn CxSetup.exe, chọn start



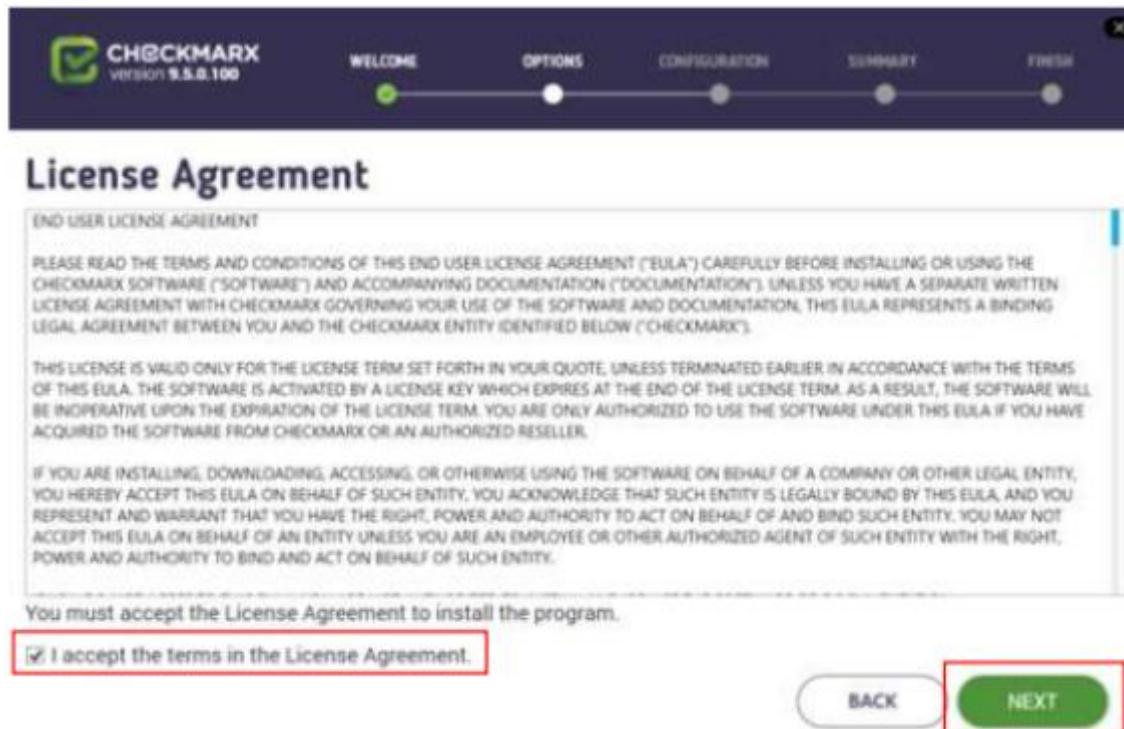
Hình 21. Cài đặt Checkmarx

- Chọn ALL-IN\_ONE INSTALLATION



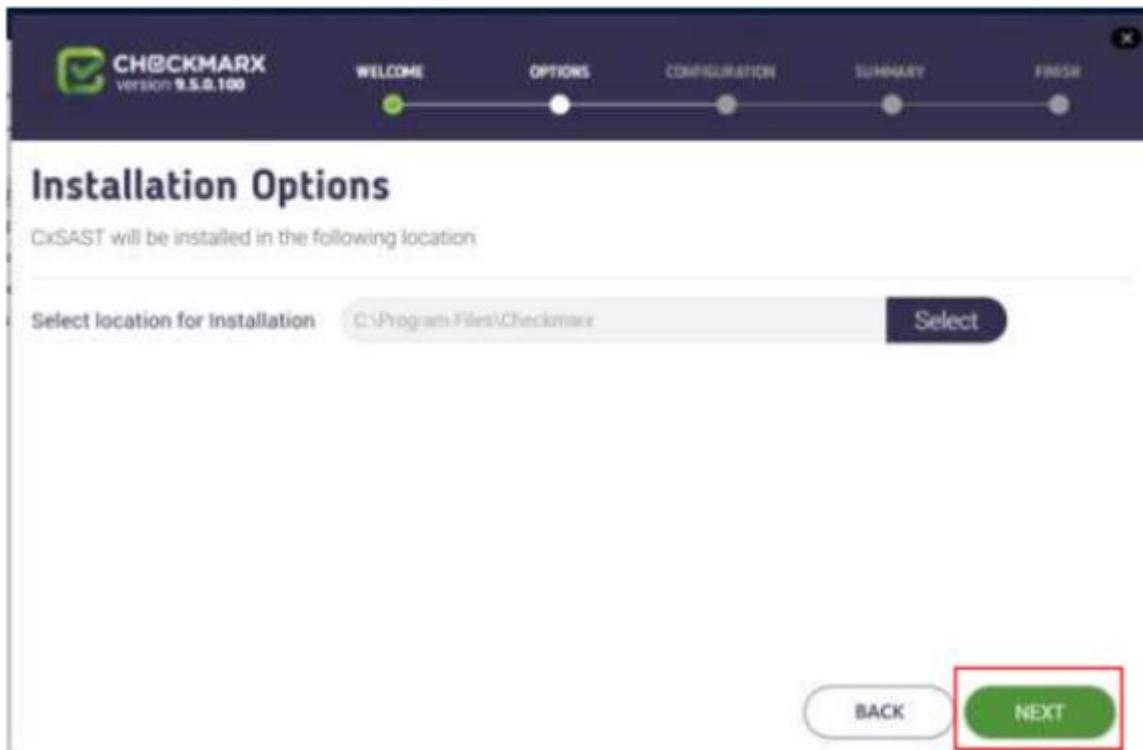
Hình 22. Cài đặt Checkmarx

- Nhấn Next



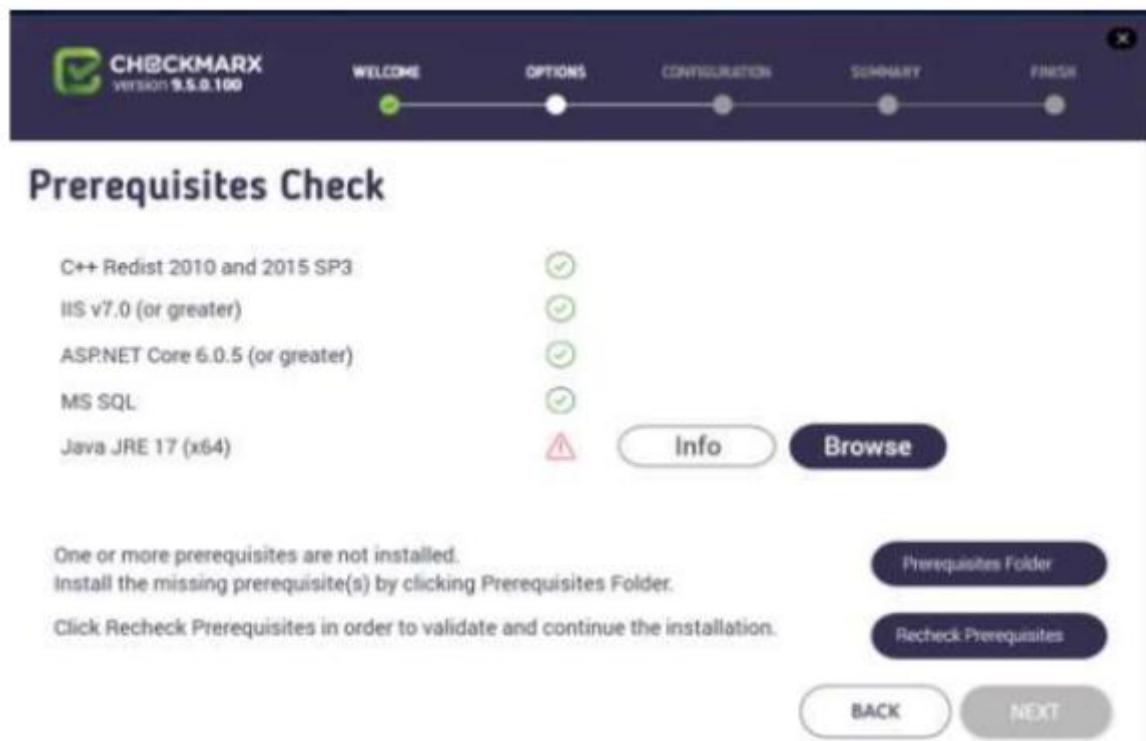
Hình 23. Cài đặt Checkmarx

- Nhấn Next



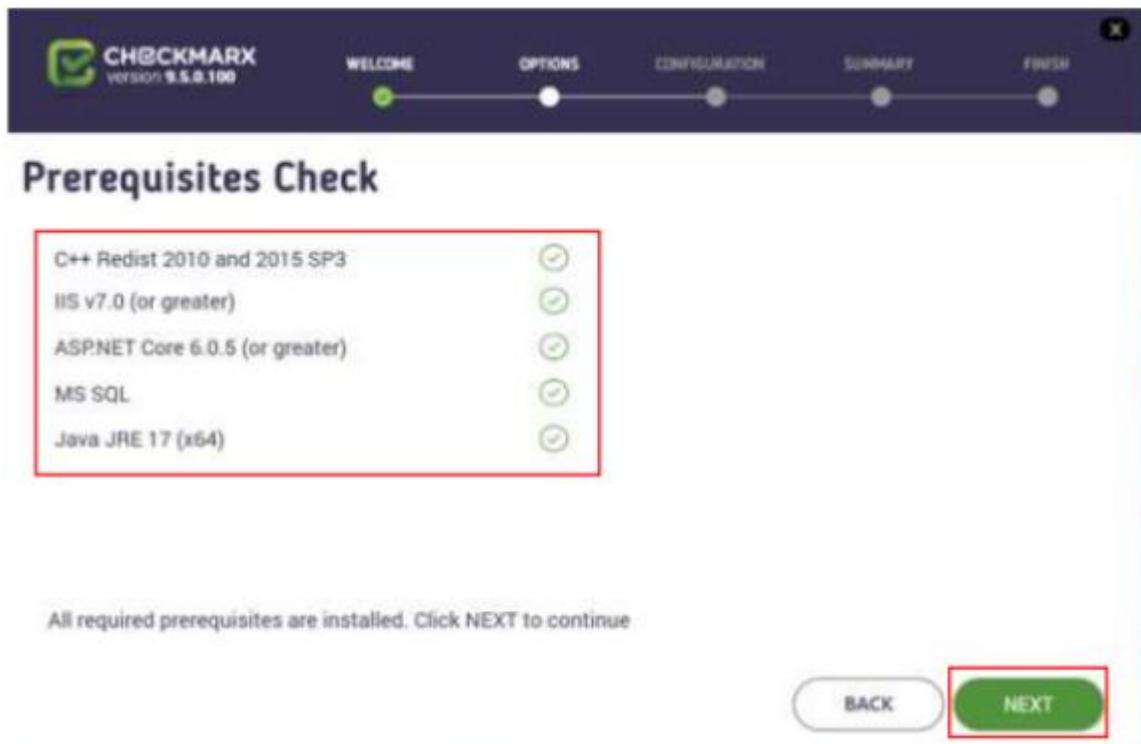
Hình 24. Cài đặt Checkmarx

- Sẽ có thông báo lỗi vì bản Java chưa được cài đặt, chọn Browse và đưa folder vào



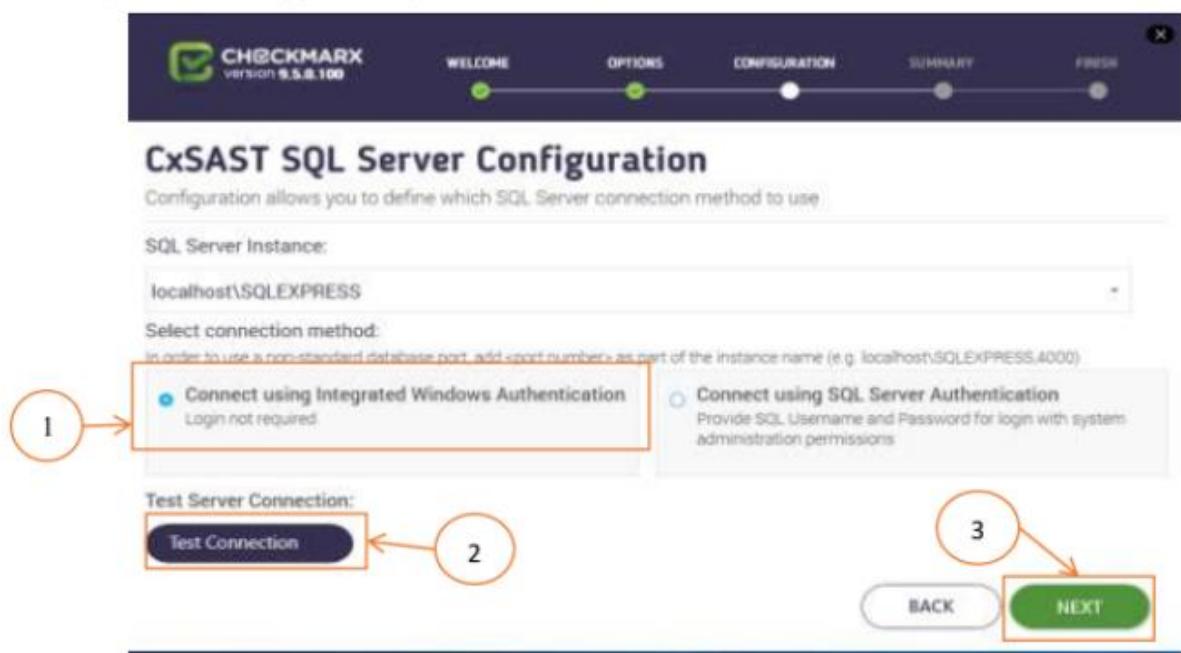
Hình 25. Cài đặt Checkmarx

- Recheck Prerequisites để kiểm tra xem các yêu cầu đã hoàn thành chưa



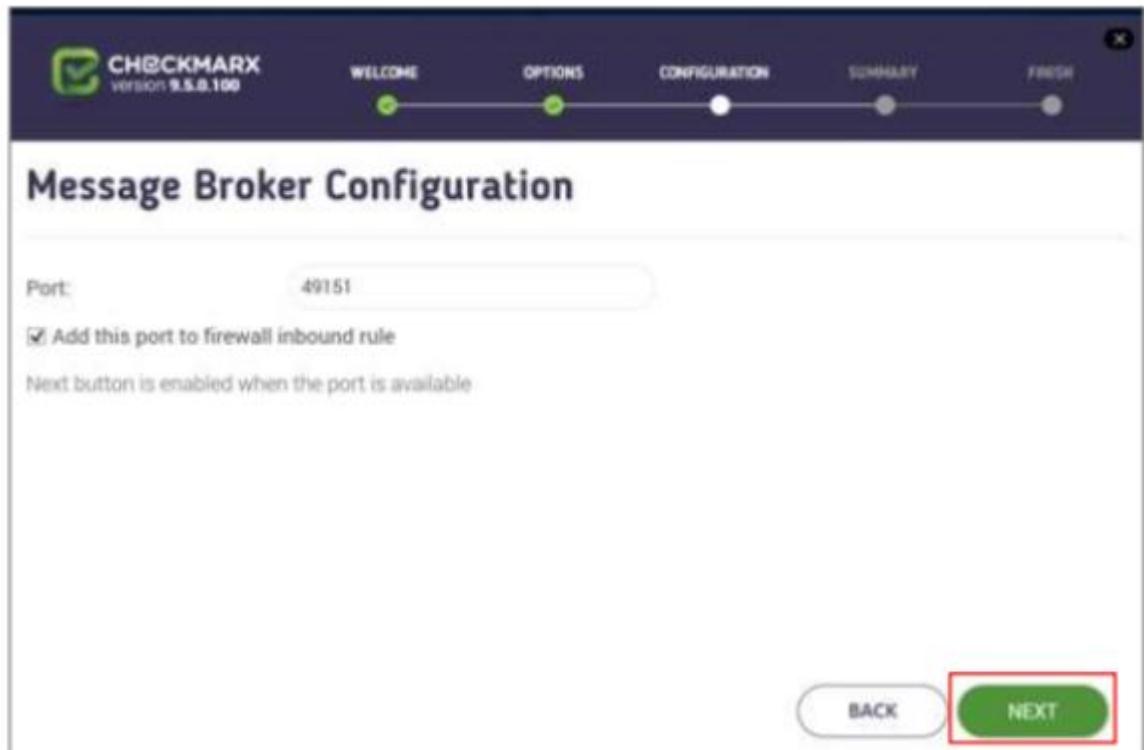
Hình 26. Cài đặt Checkmarx

- Chọn theo thứ tự bên dưới rồi nhấn Next



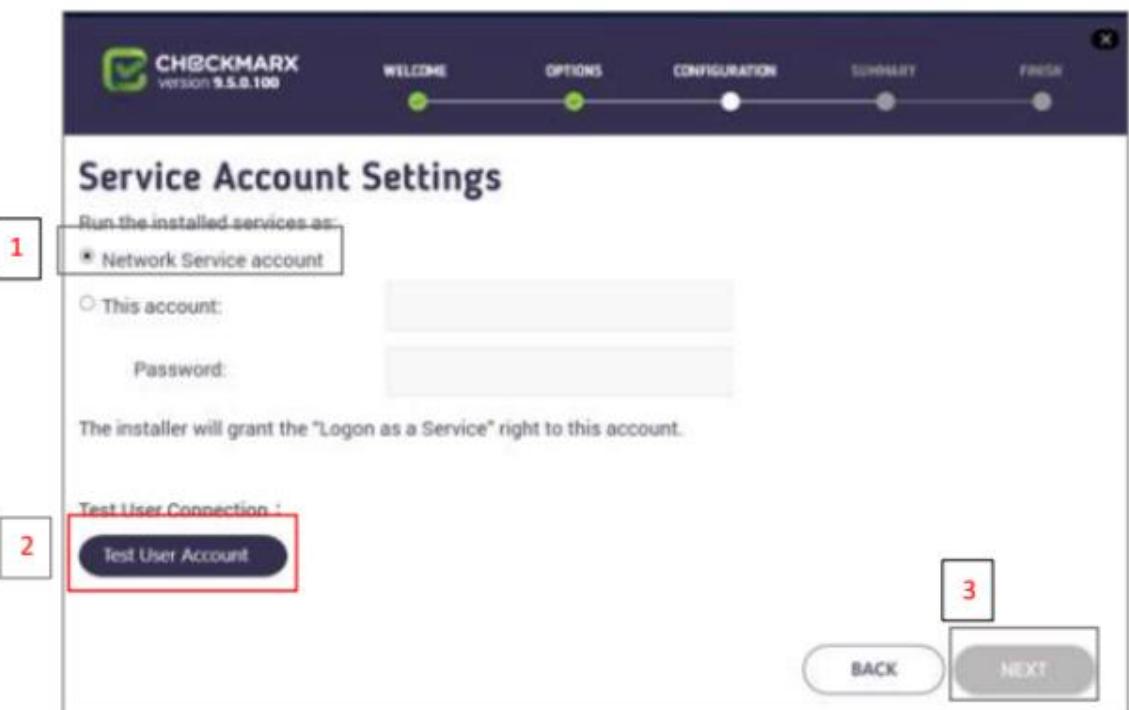
Hình 27. Cài đặt Checkmarx

- Nhấn Next



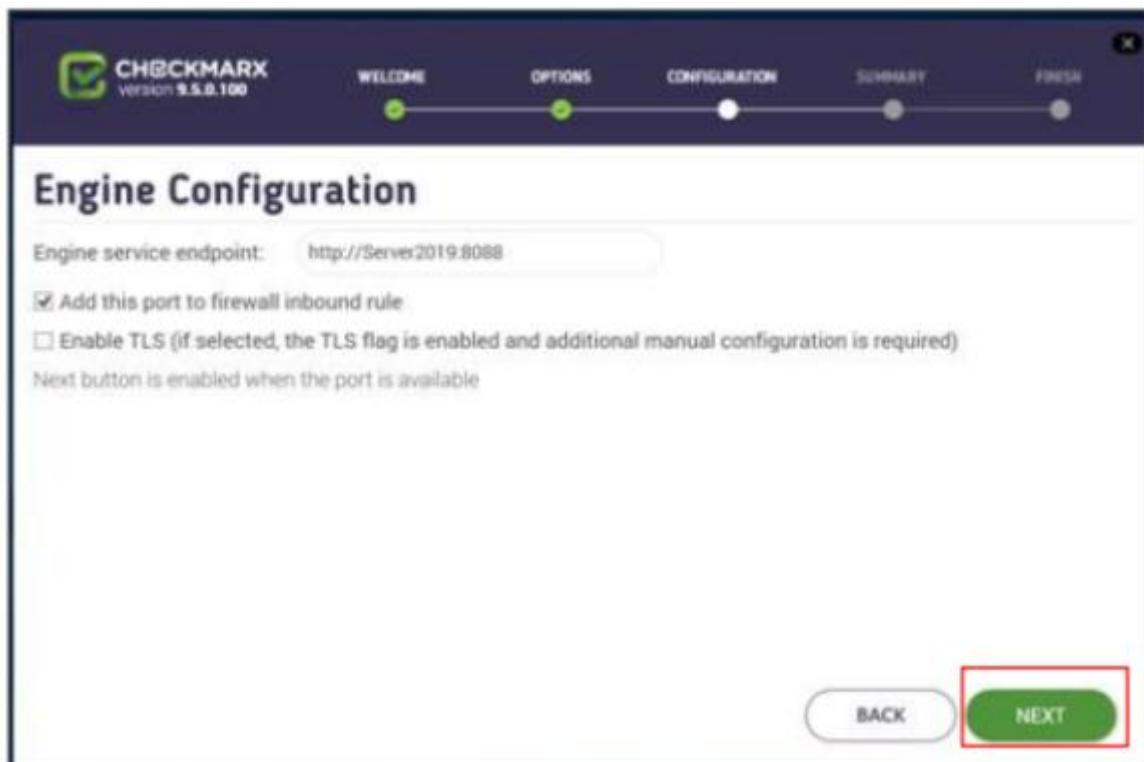
Hình 28. Cài đặt Checkmarx

- Nhấn chọn Network Service account -> Test User Account -> Next



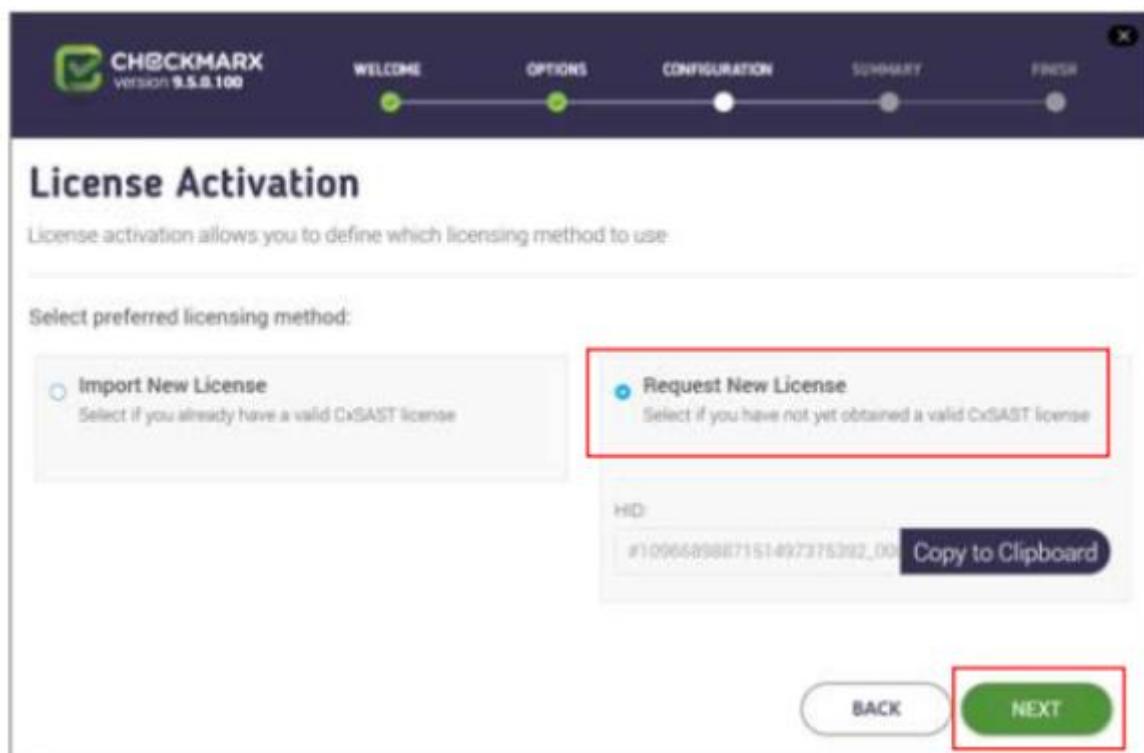
Hình 29. Cài đặt Checkmarx

- Nhấn Next



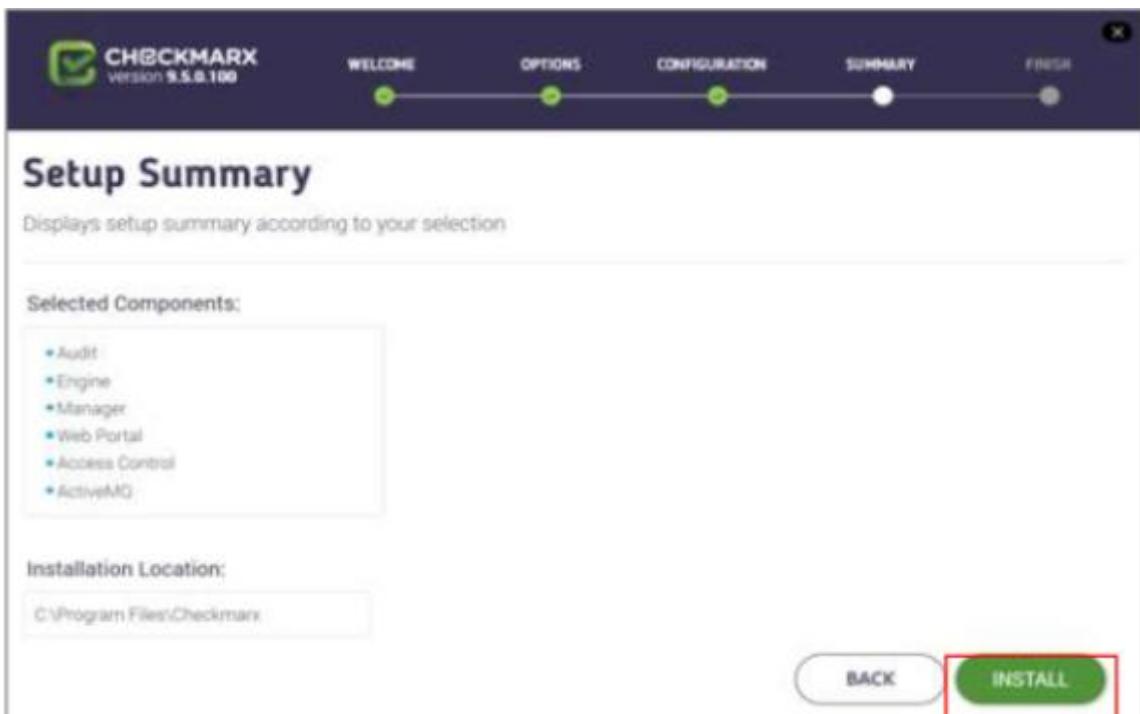
Hình 30. Cài đặt Checkmarx

- Chọn Request New License rồi nhấn Next

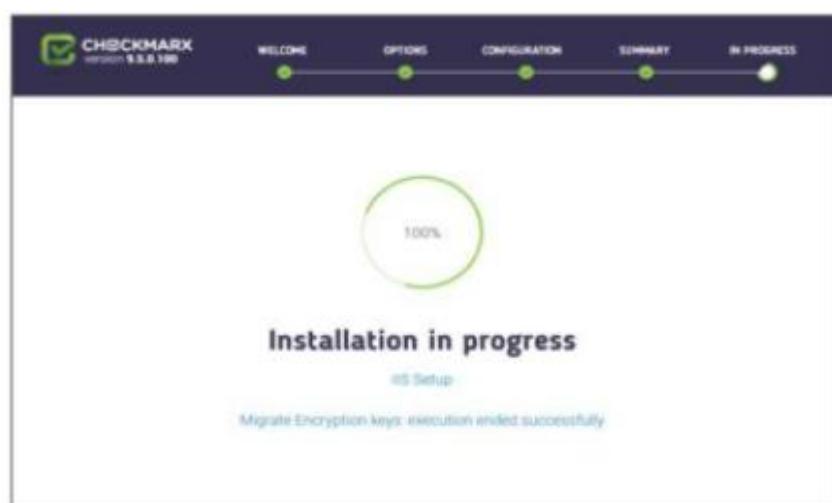


Hình 31. Cài đặt Checkmarx

- Nhấn Install



Hình 32. Cài đặt Checkmarx



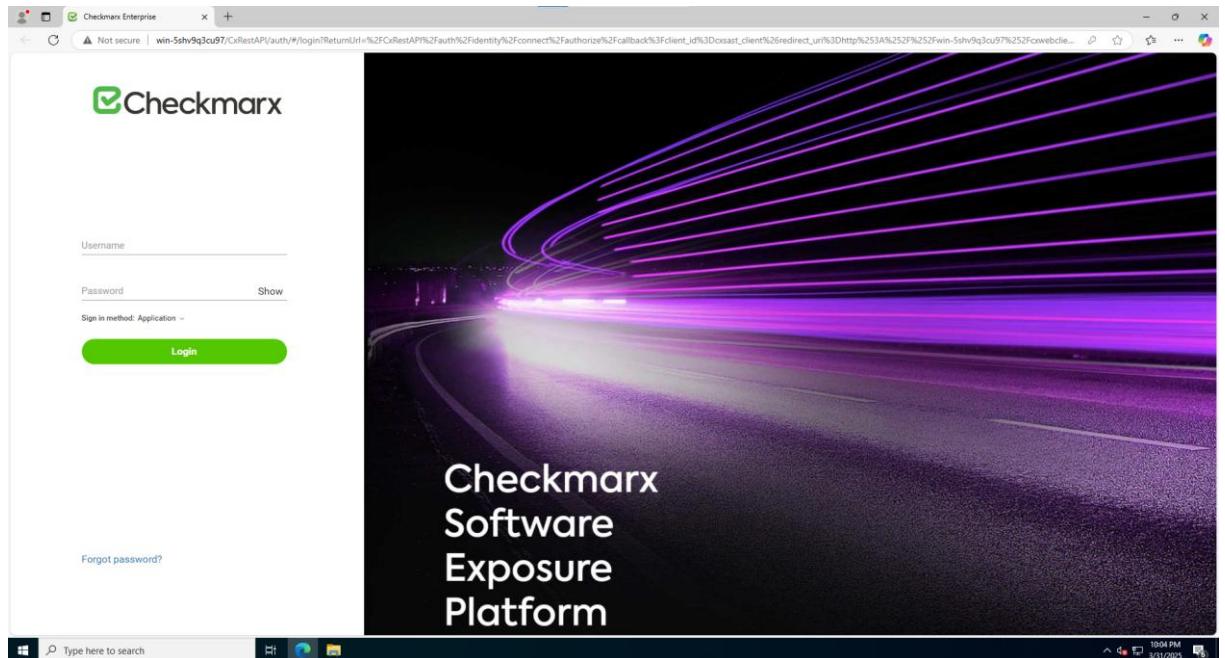
Hình 33. Cài đặt Checkmarx thành công

- Sau khi install, vào folder crack và chạy nó lên



Hình 34. Chạy folder crack

- Sau khi crack, chọn icon Checkmarx Portal để vào trang web của Checkmarx



Hình 35. Màn hình đăng nhập Checkmarx

## 2. Triển khai đề tài với checkmarx

### 2.1. Tạo Project với checkmarx

- Tại Project&Scan, chọn Create New Project

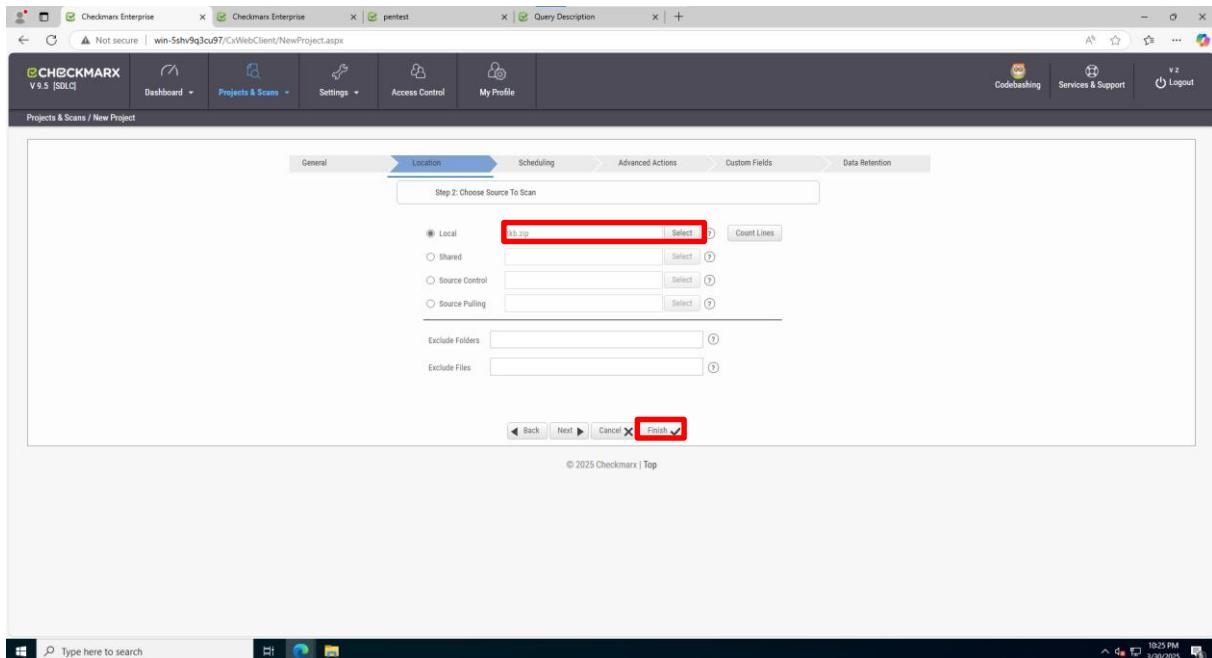
The screenshot shows the Checkmarx Enterprise web interface. At the top, there are several tabs: 'Checkmarx Enterprise' (active), 'Checkmarx Enterprise', 'pentest', and 'Query Description'. Below the tabs is a navigation bar with icons for Dashboard, Projects & Scans (highlighted), Settings, Access Control, and My Profile. On the far right are links for CodeBashing, Services & Support, and Logout. The main content area is titled 'Dashboard / Projects State'. A red box highlights the 'Create New Project' button in the top right corner of the 'Projects' tab. Below it, a table lists two projects: 'pentest01' and 'pentest'. The table includes columns for PROJECT NAME, LAST SCAN, TEAM, LOC, RISK LEVEL SCORE, HIGH VULNERABILITIES, MEDIUM VULNERABILITIES, and ACTIONS. At the bottom left is a page navigation bar with arrows and a page size dropdown set to 10. At the bottom right, it says '2 items in 1 pages'. The footer contains the text '© 2025 Checkmarx | Top'.

Hình 36. Tạo Project

The screenshot shows the 'New Project' setup screen. At the top, there are tabs for General, Location, Scheduling, Advanced Actions, Custom Fields, and Data Retention. The 'General' tab is active and highlighted with a blue arrow. Below it, a sub-section titled 'Step 1: Enter Project General Settings' is shown. A red box highlights the 'Project Name' input field, which contains the value 'pentest01'. Other fields include 'Preset' (set to 'Checkmarx Default'), 'Configuration' (set to 'Default Configuration'), and 'Team' (set to 'CxServer'). At the bottom are buttons for 'Back', 'Next >', 'Cancel X', and 'Finish ✓'.

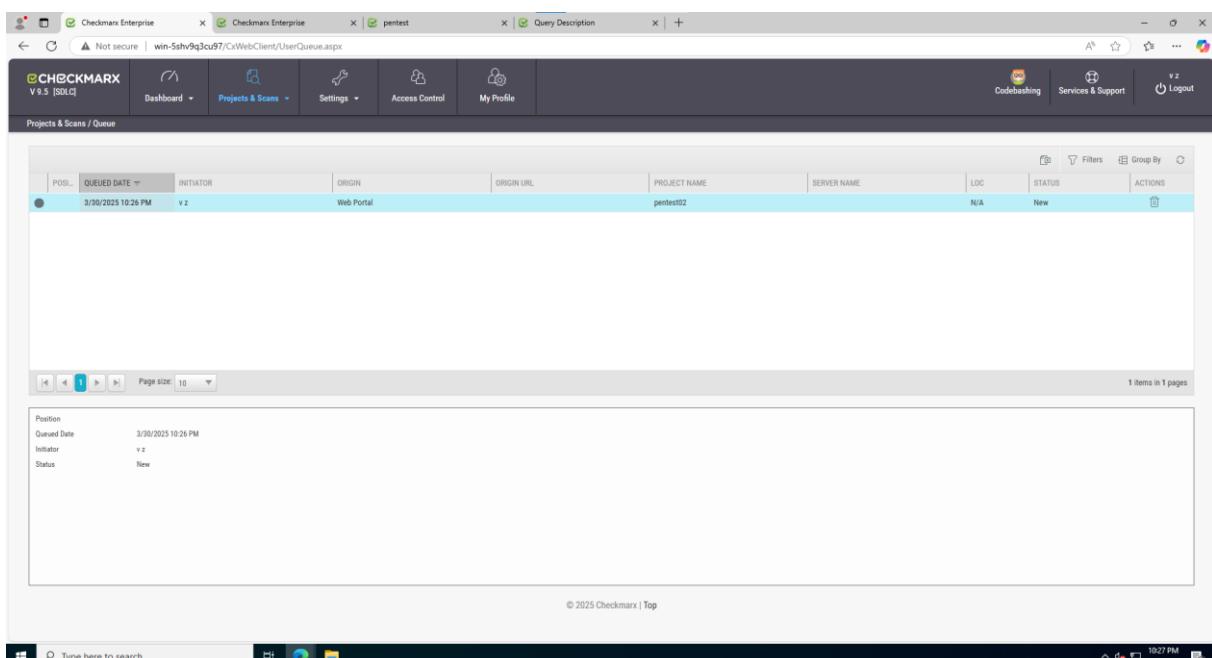
Hình 37. Đặt tên Project

- Tại Local, chọn Select để upload file cần quét



Hình 38. Upload file cần scan lỗi

- Nhấn Finish và chờ quét



Hình 39. Màn hình hiển thị đang trong quá trình scan lỗi

- Đã Scan thành công

The screenshot shows the Checkmark Enterprise web application. At the top, there are four tabs: 'Checkmark Enterprise' (active), 'Checkmark Enterprise', 'pen test', and 'Query Description'. Below the tabs is a dark header bar with the 'CHECKMARK V 9.5 [SQLC]' logo, navigation links ('Dashboard', 'Projects & Scans', 'Settings', 'Access Control', 'My Profile'), and user information ('Codebashing', 'Services & Support', 'Logout'). The main content area is titled 'Projects & Scans / Queue'. It displays a table with one item:

Position	QUEUED DATE	INITIATOR	ORIGIN	ORIGIN URL	PROJECT NAME	SERVER NAME	LOC	STATUS	ACTIONS
1	3/30/2025 10:26 PM	v z	Web Portal		pen test02	localhost	212	Finished	

Below the table, there's a message: 'Scan completed'. At the bottom right of the page, it says '1 items in 1 pages'. The footer includes the copyright notice '© 2025 Checkmark | Top' and a Windows taskbar at the bottom.

Hình 40. Màn hình scan lỗi thành công

- Tại Dashboard, nhấn Project State để xem kết quả

This screenshot is similar to the previous one but shows the 'Project State' tab selected in the navigation bar. The main content area is titled 'Projects & Scans / Queue' and shows the same table with one item. The message 'Scan completed' is present below the table. The footer and taskbar are identical to the previous screenshot.

Hình 41. Xem kết quả các file đã scan tại đây.

The screenshot shows the Checkmarx Enterprise web interface. At the top, there's a navigation bar with tabs for 'Dashboard', 'Projects & Scans', 'Settings', 'Access Control', and 'My Profile'. On the right side of the header are links for 'CodeBashing', 'Services & Support', and 'Logout'. Below the header is a main content area titled 'Dashboard / Projects State'. It displays a table with columns: PROJECT NAME, LAST SCAN DATE, TEAM, LOC, RISK LEVEL SCORE, HIGH VULNERABILITIES, MEDIUM VULNERABILITIES, and ACTIONS. The table contains three rows corresponding to the projects listed above. At the bottom of the table, there are navigation buttons for 'Page size' (set to 10) and '3 items in 1 pages'. A copyright notice '© 2025 Checkmarx | Top' is at the bottom.

Hình 42. Danh sách các file đã được scan thành công

## 2.2. Phân tích lỗi

- Chọn Project đã tạo

The screenshot shows the 'Project State / pentest02' page. At the top, there's a navigation bar with tabs for 'Dashboard', 'Projects & Scans', 'Settings', 'Access Control', and 'My Profile'. On the right side of the header are links for 'CodeBashing', 'Services & Support', and 'Logout'. Below the header is a main content area titled 'Project State / pentest02'. It displays several sections: 'Current status (Public Scan on 3/30/2025 10:27:26 PM)', 'SAST Vulnerabilities Status' (with counts for High, Med, Low, and Recurrent vulnerabilities), 'SAST progress status' (a chart showing the distribution of vulnerabilities across severity levels), and 'Open Source Analysis (OSA)' (status message: 'Open Source Analysis for this project has not yet been performed'). At the bottom, there are buttons for 'Full Scan', 'Incremental Scan', 'Run OSA', and 'Actions'. A copyright notice '© 2025 Checkmarx' is at the very bottom.

Hình 43. Thông tin Project

Ảnh trên cho biết:

## 1. Thông tin dự án

- **Tên dự án:** PENTEST02

- **Trạng thái quét:** Quét công khai được thực hiện vào 30/03/2025 lúc 10:27:26 PM

## 2. Trạng thái lỗ hổng SAST (SAST Vulnerabilities Status)

- Hiển thị số lượng lỗ hổng tìm thấy theo mức độ nghiêm trọng:

- 1 lỗ hổng nghiêm trọng (High)
- 2 lỗ hổng trung bình (Medium)
- 10 lỗ hổng thấp (Low)
- Không có lỗ hổng tái diễn hoặc đã được giải quyết

## 3. Tiến trình quét SAST (SAST Progress Status)

- Biểu đồ cột cho thấy số lượng lỗ hổng đã phát hiện theo từng mức độ.
- Hiện tại, 1 High, 2 Medium, 10 Low vẫn còn tồn tại.

## 4. Phân tích mã nguồn mở (Open Source Analysis - OSA)

- Chưa thực hiện quét OSA để kiểm tra lỗ hổng trong các thư viện nguồn mở.
- Có thể có thư viện dễ bị tấn công hoặc đã lỗi thời.

## 5. Các tùy chọn quét

- **Full Scan:** Quét toàn bộ mã nguồn.
- **Incremental Scan:** Quét những thay đổi kể từ lần quét trước.
- **Run OSA:** Thực hiện quét OSA để phân tích thư viện nguồn mở.

## Nhận xét

- Dự án có một lỗ hổng nghiêm trọng, cần được xem xét ngay.
- Chưa thực hiện phân tích thư viện nguồn mở, nên cần chạy OSA để đảm bảo không có lỗ hổng trong các dependencies.
- Dự án có thể tiếp tục cải thiện bằng cách sửa lỗi và chạy các lần quét mới.

- Tại Action bên trên góc phải, ấn tùy chọn Open Viewer để xem code

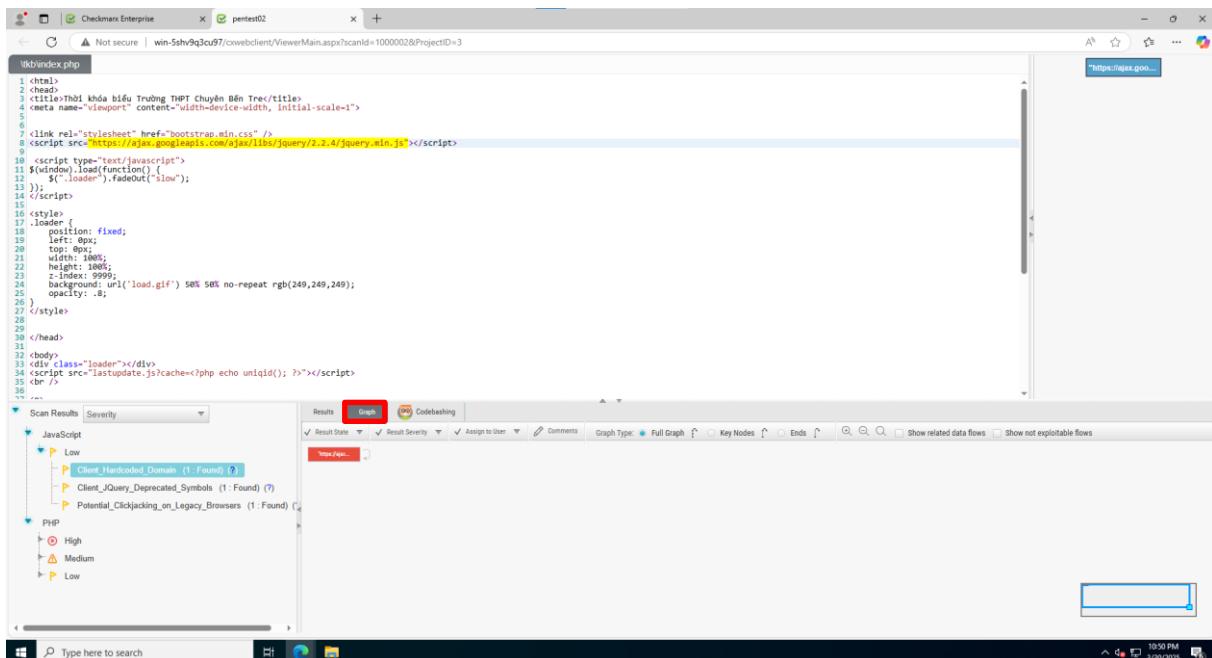
The screenshot shows the Checkmark Enterprise web interface. In the top right corner, there is a dropdown menu with several options. One of the options, 'Open Viewer', is highlighted with a red rectangle.

Hình 44. Nhấn tùy chọn Open Viewer

The screenshot shows the 'Scan Results' section of the Checkmark Enterprise web interface. It lists various vulnerabilities categorized by severity (High, Medium, Low). A specific entry is highlighted, showing a detailed code review of a file named 'vkb/admin/index.php'. The code snippet shows a path traversal vulnerability where user input is used to construct a file path.

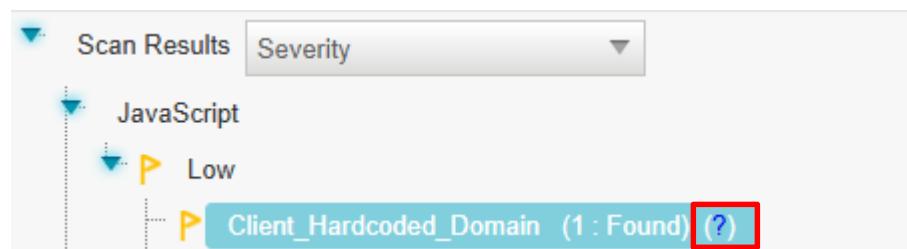
Hình 45. Màn hình hiển thị sau khi ấn Open Viewer

- Nhấn vào Graph để xem cách fix lỗi hỏng nhanh nhất bằng các liệt kê các đường input, output



Hình 46. Màn hình hiển thị sau khi nhấn Graph

- Có thể ánh vào biểu tượng dấu chấm hỏi bên cạnh lỗi đó để có thể hiểu đó là lỗi gì, nguyên nhân và cách khắc phục.



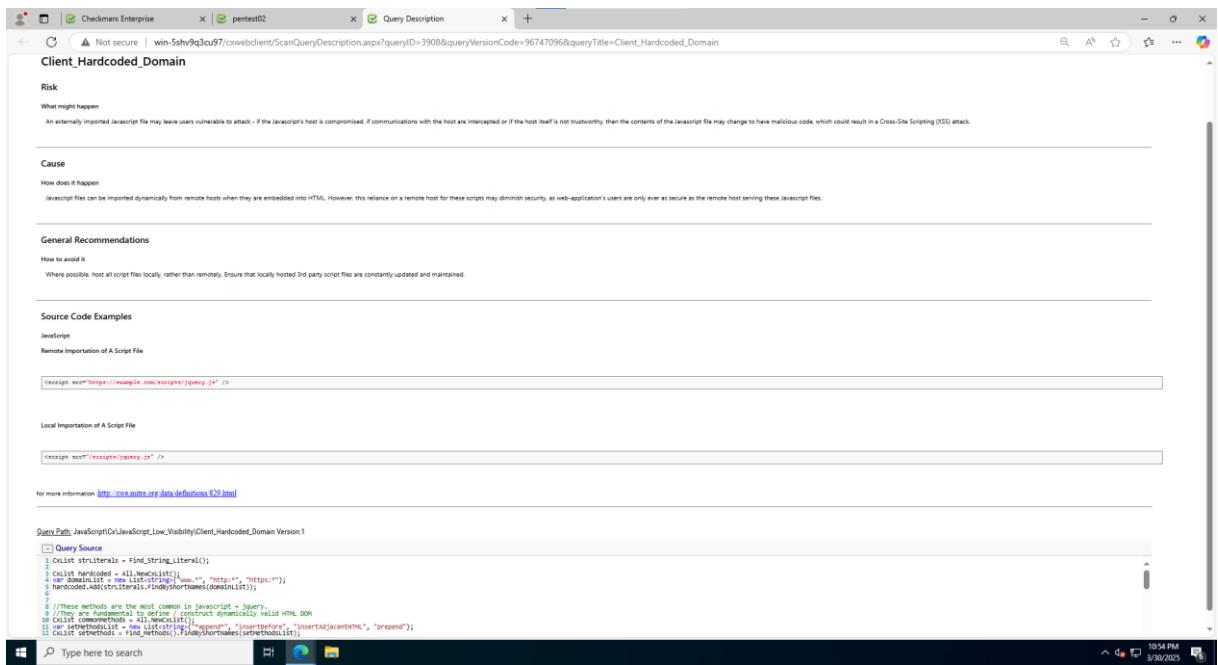
Hình 47. Xem chi tiết và cách fix lỗi tại đây

## 2.3. Phân tích từng lỗi có trên Project

### 2.3.1. JavaScript

#### ⊕ Low

❖ **Client\_Hardcoded\_Domain:** Lỗi hỏng bảo mật từ Checkmarx Enterprise



Hình 48. Client\_Hardcoded\_Domain: Lỗi hỏng bảo mật từ Checkmarx Enterprise

#### 1. Rủi ro (Risk)

- Một tệp JavaScript được nhập từ bên ngoài có thể khiến ứng dụng dễ bị tấn công.
- Nếu máy chủ từ xa bị tấn công hoặc không đáng tin cậy, mã JavaScript có thể bị thay đổi thành mã độc hại.
- Điều này có thể dẫn đến tấn công Cross-Site Scripting (XSS).

#### 2. Nguyên nhân (Cause)

- Các tệp JavaScript có thể được nhập động từ các máy chủ từ xa khi nhúng vào HTML.
- Điều này có thể làm giảm mức độ bảo mật, vì ứng dụng web chỉ an toàn khi máy chủ từ xa đáng tin cậy.

### **3. Khuyến nghị chung (General Recommendations)**

- Lưu trữ các tệp JavaScript cục bộ thay vì sử dụng từ xa.
- Đảm bảo rằng các tệp JavaScript được cập nhật và duy trì thường xuyên.

### **4. Ví dụ mã nguồn (Source Code Examples)**

- Import từ xa (Để bị tấn công):

```
<script src="https://example.com/scripts/jquery.js" />
```

*Hình 49. Lệnh import từ xa*

- Import cục bộ (Được khuyến nghị):

```
<script src="/scripts/jquery.js" />
```

*Hình 50. Lệnh import cục bộ*

### **5. Nguồn truy vấn (Query Source)**

- Đoạn mã phân tích cú pháp các chuỗi liên quan đến miền (domain) và phương thức HTML.
- Nó xác định các phương thức có thể thêm nội dung động vào DOM (append, insertBefore, prepend...), có thể bị khai thác.

### **6. Kết luận**

- Vấn đề chính: Nhúng JavaScript từ một miền cố định bên ngoài có thể làm ứng dụng dễ bị tấn công XSS.
- Cách khắc phục: Sử dụng JavaScript được lưu trữ cục bộ thay vì phụ thuộc vào máy chủ bên ngoài.

- ❖ **Client\_JQuery\_Deprecated\_Symbols:** Các vấn đề bảo mật liên quan đến việc sử dụng các biểu tượng/hàm jQuery đã bị loại bỏ.

The screenshot shows a browser window with multiple tabs. The active tab is titled "Query Description" and contains the following content:

- Client\_JQuery\_Deprecated\_Symbols**
- Risk**
  - What might happen**: Referencing deprecated modules can cause an application to be exposed to known vulnerabilities, that have been publicly reported and already fixed. A common attack technique is to scan applications for these known vulnerabilities, and then exploit the application through these deprecated versions. However, even if deprecated code is used in a way that is completely secure, its very use and inclusion in the code base would encourage developers to re-use the deprecated element in the future, potentially leaving the application vulnerable to attack, which is why deprecated code should be eliminated from the code-base as a matter of practice.
  - Note**: That the actual risk involved depends on the specifics of any known vulnerabilities in older versions.
  - Use**: Use of a deprecated API on client code may leave users vulnerable to browser-based attacks; this is exacerbated by the fact client-side code is available to any attacker with client access, who may be able to trivially detect use of this deprecated API.
- Cause**
  - How does it happen**: The application references code elements that have been declared as deprecated. This could include classes, functions, methods, properties, modules, or obsolete library versions that are either out of date by version, or have been entirely deprecated. It is likely that the code that references the obsolete element was developed before it was declared as obsolete, and in the meantime the referenced code was updated.
- General Recommendations**
  - How to avoid it**: Always prefer to use the most updated versions of libraries, packages, and other dependencies.
  - Do not use or reference any class, method, function, property, or other element that has been declared deprecated.
- Source Code Examples**

Hình 51. Client\_JQuery\_Deprecated\_Symbols: Các vấn đề bảo mật liên quan đến việc sử dụng các biểu tượng/hàm jQuery đã bị loại bỏ.

The screenshot shows a browser window with multiple tabs. The active tab is titled "Query Description" and contains the following content:

- Source Code Examples**
  - JavaScript**
    - jQuery - Using the Deprecated \$.parseJSON**: `$.parseJSON(json); // Legacy method for support of older browsers; tends to throw unexpected exceptions with certain control characters`
  - Using a Native Call Instead of Deprecated jQuery Calls**: `JSON.parse(json); // Native call to replace $.parseJSON(json)`
  - Obtain Year via Deprecated JavaScript Method**: `var d = new Date(); var year = d.getYear(); // getYear() is deprecated and affected by Y2K; for a given year, 20xx, it will return 1xx.`
  - Obtain Year via a Supported JavaScript Method**: `var d = new Date(); var year = d.getFullYear();`
  - Invoking a Deprecated Function. Denoted Using JSDoc**: `/** @deprecated */ function myOldFunction() { /* Code that is deprecated */ }`

Hình 52. Client\_JQuery\_Deprecated\_Symbols: Các vấn đề bảo mật liên quan đến việc sử dụng các biểu tượng/hàm jQuery đã bị loại

The screenshot shows a browser window with several tabs open, including 'Thực hiện Scan code sử dụng Ch...'. The main content area displays a security analysis for 'Client\_JQuery\_Deprecated\_Symbols'. It includes sections for 'Obtain Year via a Supported JavaScript Method' and 'Invoking a Deprecated Function, Denoted Using JSDoc'. A code snippet for 'myOldFunction()' is shown with a note about it being deprecated. Below this, a link to 'http://cve.mitre.org/data/definitions/477.html' is provided. At the bottom, there's a code editor window titled 'Query Path: JavaScript!CxJavaScript\_Low\_Visibility!Client\_JQuery\_Deprecated\_Symbols Version:1' containing the source code for 'jQuery1.7.1.js'.

*Hình 53. Client\_JQuery\_Deprecated\_Symbols: Các vấn đề bảo mật liên quan đến việc sử dụng các biểu tượng/hàm jQuery đã bị loại*

## 1. Rủi ro (Risk)

- Lỗ hổng bảo mật:
  - Sử dụng các module hoặc API jQuery đã bị deprecated có thể khiến ứng dụng dễ bị tấn công bảo mật, vì chúng có thể chứa các lỗ hổng đã biết nhưng không còn được cập nhật.
  - Hacker có thể quét ứng dụng để tìm các lỗ hổng đã biết và khai thác chúng.
- Không tương thích với jQuery mới:
  - Khi nâng cấp lên phiên bản jQuery mới, các hàm deprecated có thể không còn hoạt động, gây lỗi ứng dụng.
- Rủi ro lan truyền lỗi:
  - Mã chứa các phương thức deprecated có thể khiến lập trình viên mới vô tình tái sử dụng, làm tăng nguy cơ bảo mật trong tương lai.
- Dễ bị phát hiện và khai thác:
  - Vì JavaScript chạy phía client, kẻ tấn công có thể dễ dàng kiểm tra mã nguồn để tìm các API jQuery lỗi thời và thực hiện các cuộc tấn công dựa trên trình duyệt.

## 2. Nguyên nhân (Cause)

- Úng dụng vẫn sử dụng các phương thức hoặc thư viện jQuery cũ đã bị loại bỏ (deprecated).
- Có thể do code được viết từ trước, khi những API này vẫn còn hợp lệ.
- Không kiểm tra và cập nhật thư viện jQuery theo thời gian.

## 3. Khuyến nghị chung (General Recommendations)

- Luôn ưu tiên sử dụng các phiên bản cập nhật nhất của thư viện, gói và các phụ thuộc khác.
- Không sử dụng hoặc tham chiếu đến bất kỳ lớp, phương thức, hàm, thuộc tính hoặc phần tử nào khác đã được khai báo là bị loại bỏ.

## 4. Ví dụ mã nguồn (Source Code Examples)

- jQuery - Sử dụng `$.parseJSON` bị loại bỏ

```
$.parseJSON(json); // Legacy method for support of older browsers; tends to throw unexpected exceptions with certain control characters
```

Hình 54. jQuery - Sử dụng `$.parseJSON` bị loại bỏ

- Sử dụng Native Call thay vì jQuery Calls bị loại bỏ

```
JSON.parse(json); // Native call to replace $.parseJSON(json)
```

Hình 55. Sử dụng Native Call thay vì jQuery Calls bị loại bỏ

- Lấy năm qua phương thức JavaScript bị loại bỏ

```
var d = new Date();
var year = d.getYear(); // getYear() is deprecated and affected by Y2K; for a given year, 20xx, it will return 1xx.
```

Hình 56. Lấy năm qua phương thức JavaScript bị loại bỏ

- Lấy năm qua phương thức JavaScript được hỗ trợ

```
var d = new Date();
var year = d.getFullYear();
```

Hình 57. Lấy năm qua phương thức JavaScript được hỗ trợ

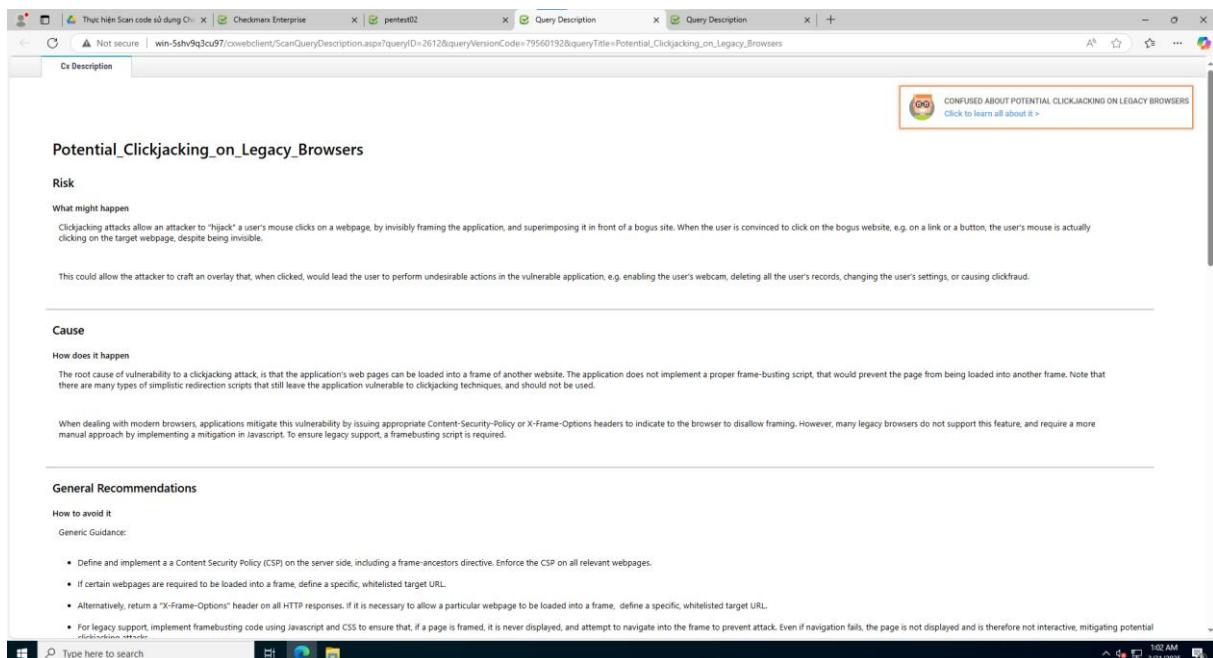
- Gọi một hàm bị loại bỏ, được chú thích bằng JSDoc

```
/** @deprecated */
function myOldFunction() {
    /* Code that is deprecated */
}

myOldFunction();
```

Hình 58. Gọi một hàm bị loại bỏ, được chú thích bằng JSDoc

❖ **Potential\_Clickjacking\_on\_Legacy\_Browsers:** Lỗi hổng Clickjacking trên các trình duyệt cũ (Legacy Browsers)



Hình 59. Potential\_Clickjacking\_on\_Legacy\_Browsers: Lỗi hổng Clickjacking trên các trình duyệt cũ (Legacy Browsers)

**Specific Recommendations:**

- Implement a proper framebuster script on the client, that is not vulnerable to frame-buster-busting attacks.
  - Code should first disable the UI, such that even if frame-busting is successfully evaded, the UI cannot be clicked. This can be done by setting the CSS value of the "display" attribute to "none" on either the "body" or "html" tags. This is done because, if a frame attempts to redirect and become the parent, the malicious parent can still prevent redirection via various techniques.
  - Code should then determine whether no framing occurs by comparing self == top; if the result is true, can the UI be enabled. If it is false, attempt to navigate away from the framing page by setting the top.location attribute to self.location.

**Source Code Examples**

**JavaScript**

**Clickjammable Webpage**

```
<html>
<body>
<button onclick="clicked()">
  Click here if you love ducks
</button>
</body>
</html>
```

**Bustable Framebuster**

```
<html>
<head>
<script>
if ( window.self.location != window.top.location ) {
  window.top.location = window.self.location;
}
</script>
</head>
<body>
<button onclick="clicked()">
  Click here if you love ducks
</button>
</body>
</html>
```

*Hình 60. Potential\_Clickjacking\_on\_Legacy\_Browsers: Lỗ hổng Clickjacking trên các trình duyệt cũ (Legacy Browsers)*

**Bustable Framebuster**

```
<html>
<head>
<script>
if ( window.self.location != window.top.location ) {
  window.top.location = window.self.location;
}
</script>
</head>
<body>
<button onclick="clicked()">
  Click here if you love ducks
</button>
</body>
</html>
```

**Proper Framebusterbusting**

```
<html>
<head>
<style> html {display : none;} </style>
<script>
  if ( self === top ) {
    document.documentElement.style.display = 'block';
  }
  else {
    top.location = self.location;
  }
</script>
</head>
<body>
<button onclick="clicked()">
  Click here if you love ducks
</button>
</body>
</html>
```

for more information <http://cve.mitre.org/data/definitions/693.html>

Query Path: JavaScript\JavaScript\_Low\_Visibility\Potential\_Clickjacking\_on\_Legacy\_Browsers Version:1

Query Source

```
1 result = Find_Insufficient_ClickJacking_Protection_OnClient();
```

*Hình 61. Potential\_Clickjacking\_on\_Legacy\_Browsers: Lỗ hổng Clickjacking trên các trình duyệt cũ (Legacy Browsers)*

```
Query Path: JavaScript\Cx\JavaScript_Low_Visibility\Potential_Clickjacking_on_Legacy_Browsers Version:1
[+] Query Source
1 result = Find_Insufficient_Clickjacking_Protection_OnClient();
2
3 CxList htmlDocs = All.NewCxList();
4
5 //Exclude all results if have a Lightning or LightningWebComponents project
6 //((this vulnerability is saved from the server - side)
7 if(!cxScan.isFrameworkActive("Lightning") & !cxScan.isFrameworkActive("LightningWebComponents")) {
8     htmlDocs = All.FindByRegexExt("*.html");
9     htmlDocs.Add(All.FindByShortName("CxJSNS").GetAncestorsOfType<NamespaceDecl>());
10
11     string[] unwantedExtensions = new string[] {
12         ".js", ".xss", ".xss10", ".xsaccess", ".xsapp", ".xml", ".json", ".ts", ".tsx" };
13     htmlDocs += htmlDocs.FindByFileNames(unwantedExtensions);
14 }
```

Hình 62. Potential\_Clickjacking\_on\_Legacy\_Browsers: Lỗi hổng Clickjacking trên các trình duyệt cũ (Legacy Browsers)

### 1. Rủi ro (Risk)

- Tấn công Clickjacking cho phép kẻ tấn công chiếm quyền kiểm soát các cú nhấp chuột của người dùng trên một trang web.
- Kẻ tấn công chèn trang web hợp lệ vào một iframe ẩn, sau đó đặt nó trên một trang web giả mạo.
- Khi người dùng nhấp vào trang web giả mạo, họ vô tình thực hiện hành động trên trang web thực tế mà họ không biết.
- Hậu quả có thể bao gồm:
  - Bật webcam mà không có sự đồng ý.
  - Xóa dữ liệu của người dùng.
  - Thay đổi cài đặt tài khoản.
  - Click fraud (tấn công lừa đảo bằng nhấp chuột).

### 2. Nguyên nhân (Cause)

- Trang web có thể bị tải trong một frame mà không có cơ chế bảo vệ.
- Trang web không sử dụng Frame-Busting Script để ngăn bị tải trong iframe.
- Các trình duyệt hiện đại có thể bảo vệ bằng CSP (Content Security Policy) hoặc X-Frame-Options.
- Tuy nhiên, trình duyệt cũ không hỗ trợ điều này, do đó cần có phương pháp bảo vệ thủ công bằng JavaScript.

### 3. Khuyến nghị chung (General Recommendations)

- Triển khai CSP (Content Security Policy) trên máy chủ với thuộc tính frame-ancestors để ngăn chặn nhúng iframe.
- Nếu cần sử dụng iframe, chỉ định danh sách URL hợp lệ.
- Sử dụng X-Frame-Options trong HTTP headers:
  - X-Frame-Options: DENY → Cấm mọi iframe.
  - X-Frame-Options: SAMEORIGIN → Chỉ cho phép iframe từ cùng miền.
- Hỗ trợ trình duyệt cũ bằng cách sử dụng JavaScript Framebuster Script.

### 4. Ví dụ mã nguồn (Source Code Examples)

- Web dễ bị Clickjacking

```
<html>
    <body>
        <button onclick="clicked();">
            Click here if you love ducks
        </button>
    </body>
</html>
```

Hình 63. Web dễ bị Clickjacking

- Framebuster cơ bản (nhưng có thể bị bypass)

```
<html>
    <head>
        <script>
            if ( window.self.location != window.top.location ) {
                window.top.location = window.self.location;
            }
        </script>
    </head>

    <body>
        <button onclick="clicked();">
            Click here if you love ducks
        </button>
    </body>
</html>
```

Hình 64. Framebuster cơ bản

- Framebuster bảo mật tốt hơn

```
<html>
  <head>
    <style> html {display : none; } </style>
    <script>
      if ( self === top ) {
        document.documentElement.style.display = 'block';
      }
      else {
        top.location = self.location;
      }
    </script>
  </head>

  <body>
    <button onclick="clicked()">
      Click here if you love ducks
    </button>
  </body>
</html>
```

Hình 65. Framebuster bảo mật tốt hơn

## 5. Kết luận

- Clickjacking có thể bị khai thác trên các trình duyệt cũ nếu không có biện pháp bảo vệ.
- Biện pháp phòng chống:
  - Dùng CSP (frame-ancestors) hoặc X-Frame-Options để ngăn iframe.
  - Với trình duyệt cũ, cần JavaScript Framebuster để bảo vệ.
  - Vô hiệu hóa UI khi phát hiện trang bị nhúng trong iframe.

### 2.3.2. PHP

#### ⊕ Hight

- ❖ **File\_Manipulation:** Lỗi hổng File Manipulation có thể dẫn đến ghi đè hoặc thực thi mã độc nếu không được kiểm soát chặt chẽ

The screenshot shows a browser window with multiple tabs open, displaying a detailed security report from Checkmarx Enterprise. The main content area is titled "File\_Manipulation". It includes sections for "Risk", "Cause", "General Recommendations", and "Source Code Examples". The "Source Code Examples" section contains a code snippet in PHP that demonstrates how an attacker can perform file manipulation by setting the "action" parameter to "w" and specifying a filename under the webroot. The code also shows how to write to the file using fwrite.

```
if (isset($_GET['logname']) && isset($_GET['action'])) {
    $action = str_replace(array("\n", "\r"), '', $_GET['action']); // Remove line breaks
    $filename = $_GET['logname']; // An attacker can provide a 'logname' that is under the webroot, creating a file that would be served by the server
    $file = fopen($filename, 'a');
    fwrite($file, $action." was performed successfully.".PHP_EOL); // An attacker can set #action to "<?php passthru($_GET['c']); ?>", resulting in a basic shell
}
```

Hình 66. File\_Manipulation: Lỗi hổng File Manipulation có thể dẫn đến ghi đè hoặc thực thi mã độc nếu không được kiểm soát chặt chẽ

The screenshot shows a browser window with multiple tabs open, displaying a detailed security report from Checkmarx Enterprise. The main content area is titled "File\_Manipulation". It includes sections for "Source Code Examples" and "File Write Location Determined Restricted By Code". The "Source Code Examples" section contains a code snippet in PHP that demonstrates how an attacker can perform file manipulation by setting the "action" parameter to "w" and specifying a filename under the webroot. The code also shows how to write to the file using fwrite. Below this, there is a note about log files being restricted to a particular folder on the system.

```
if (isset($_GET['logname']) && isset($_GET['action'])) {
    $action = str_replace(array("\n", "\r"), '', $_GET['action']); // Remove line breaks
    $filename = $_GET['logname']; // An attacker can provide a 'logname' that is under the webroot, creating a file that would be served by the server
    $file = fopen($filename, 'a');
    fwrite($file, $action." was performed successfully.".PHP_EOL);
}

if (isset($_GET['logname']) && isset($_GET['action'])) {
    $action = str_replace(array("\n", "\r"), '', $_GET['action']); // Remove line breaks
    $filename = "/var/log/application/.basename($_GET['logname']); // Can create arbitrary log files, but restricted to a particular folder on the system.
    $file = fopen($filename, 'a');
    fwrite($file, $action." was performed successfully.".PHP_EOL);
}
```

for more information <http://cwe.mitre.org/data/definitions/552.html>

Query Path: PHP\_CxPHP\_High\_Risk/File\_Manipulation Version:1

```
1 CxList Include = Find_Write();
2 CxList Inputs = Find_Interactive_Inputs();
3 CxList methods = Find_Methods();
4 CxList fileAccessExcludeList = methods.FindByShortName(new List<string>()
5     { "filetime", "is_dir", "is_executable", "is_file", "is_link", "realpath", "basename", "filesize" });
6 CxList firstParam = All.GetParameters(methods.FindByShortName("fputs"), 0);
7 fileAccessExcludeList = All.GetParameters(fileAccessExcludeList);
8 include.Add(firstParam);
9 include -= fileAccessExcludeList;
```

Hình 67. File\_Manipulation: Lỗi hổng File Manipulation có thể dẫn đến ghi đè hoặc thực thi mã độc nếu không được kiểm soát chặt chẽ

## 1. Rủi ro (Risk)

- Nếu kẻ tấn công có thể kiểm soát file cần ghi, chúng có thể:
  - Ghi đè hoặc làm hỏng tệp nhạy cảm, gây ra tấn công từ chối dịch vụ (DoS).
  - Chèn mã độc vào tệp, dẫn đến thực thi mã từ xa (RCE).

## 2. Nguyên nhân (Cause)

- Ứng dụng sử dụng dữ liệu đầu vào từ người dùng để xác định tệp ghi.
- Nếu không có cơ chế kiểm soát chặt chẽ, kẻ tấn công có thể:
  - Chỉ định tệp quan trọng như /etc/passwd, config.php để ghi đè.
  - Tạo tệp PHP độc hại trong thư mục có thể thực thi mã (webroot).

## 3. Khuyến nghị chung (General Recommendations)

- Hạn chế vị trí tệp có thể ghi:
  - Chỉ cho phép ghi vào một thư mục cụ thể và được kiểm soát.
  - Sử dụng basename() để ngăn tấn công Directory Traversal (../../etc/passwd).
  - Ví dụ:

```
$filename = "/var/log/application/" . basename($_GET['logname']);
```

Hình 68. Sử dụng basename() để ngăn tấn công Directory Traversal

- Sử dụng danh sách trắng (whitelist):
  - Chỉ cho phép ghi vào tệp được định nghĩa trước.
  - Ví dụ:

```
$allowed_files = ['log1.txt', 'log2.txt'];
if (!in_array($_GET['logname'], $allowed_files)) {
    die("Unauthorized file access!");
}
```

Hình 69. Sử dụng danh sách trắng (whitelist)

## 4. Ví dụ mã nguồn (Source Code Examples)

- Mã dễ bị tấn công (Có thể bị RCE nếu attacker tải lên mã PHP)

```
if (isset($_GET['logname']) && isset($_GET['action'])) {
    $action = str_replace(array("\n", "\r"), '', $_GET['action']); // Remove line breaks
    $filename = $_GET['logname']; // An attacker can provide a 'logname' that is under the webroot, creating a file that would be served by the server
    $file = fopen($filename, 'a');
    fwrite($file, $action." was performed successfully.".PHP_EOL); // An attacker can set $action to "<?php passthru($_GET['c']); ?>", resulting in a basic shell
}
```

Hình 70. Mã dễ bị tấn công

- Mã an toàn hơn (Hạn chế vị trí ghi, tránh RCE)

```
if (isset($_GET['logname']) && isset($_GET['action'])) {
    $action = str_replace(array("\n", "\r"), '', $_GET['action']); // Remove line breaks
    $filename = "/var/log/application/".$basename($_GET['logname']); // Can create arbitrary log files, but restricted to a particular folder on the system.
    $file = fopen($filename, 'a');
    fwrite($file, $action." was performed successfully.".PHP_EOL);
}
```

Hình 71. Mã an toàn hơn

## 5. Kết luận

- File Manipulation có thể dẫn đến ghi đè tệp quan trọng hoặc thực thi mã độc.
- Cách phòng chống:
  - Hạn chế thư mục ghi (/var/log/application/).
  - Sử dụng basename() để loại bỏ đường dẫn nguy hiểm.
  - Sử dụng danh sách trắng (whitelist) cho các tệp hợp lệ.
  - Chuyển sang cơ sở dữ liệu thay vì lưu file log.

### Medium

#### ❖ Missing\_HSTS\_Header: Lỗi hỏng Missing HSTS Header

The screenshot shows a browser window with the URL `win-5shv9q3cu97/cowclient/ScanQueryDescription.aspx?queryID=5437&queryVersionCode=119183642&queryTitle=Missing_HSTS_Header`. The page content is as follows:

**Missing\_HSTS\_Header**

**Risk**  
What might happen  
Failure to set an HSTS header and provide it with a reasonable "max-age" value of at least one year may leave users vulnerable to Man-in-the-Middle attacks.

**Cause**  
How does it happen  
Many users browse to websites by simply typing the domain name into the address bar, without the protocol prefix. The browser will automatically assume that the user's intended protocol is HTTP, instead of the encrypted HTTPS protocol.

When this initial request is made, an attacker can perform a Man-in-the-Middle attack and manipulate it to redirect users to a malicious web-site of the attacker's choosing. To protect the user from such an occurrence, the HTTP Strict Transport Security (HSTS) header instructs the user's browser to disallow use of an unsecure HTTP connection to the domain associated with the HSTS header.

Once a browser that supports the HSTS feature has visited a web-site and the header was set, it will no longer allow communicating with the domain over an HTTP connection.

Once an HSTS header was issued for a specific website, the browser is also instructed to prevent users from manually overriding and accepting an untrusted SSL certificate for as long as the "max-age" value still applies. The recommended "max-age" value is for at least one year in seconds, or 31536000.

**General Recommendations**  
How to avoid it

- Before setting the HSTS header - consider the implications it may have:
  - Forcing HTTPS will prevent any future use of HTTP, which could hinder some testing
  - Disabling HSTS is not trivial, as once it is disabled on the site, it must also be disabled on the browser
- Set the HSTS header either explicitly within application code, or using web-server configurations.

Hình 72. Missing\_HSTS\_Header: Lỗi hỏng Missing HSTS Header

**General Recommendations**

**How to avoid it**

- Before setting the HSTS header - consider the implications it may have:
  - Forcing HTTPS will prevent any future use of HTTP, which could hinder some testing
  - Disabling HSTS is not trivial, as once it is disabled on the site, it must also be disabled on the browser
- Set the HSTS header either explicitly within application code, or using web-server configurations.
- Ensure the "max-age" value for HSTS headers is set to 31536000 to ensure HSTS is strictly enforced for at least one year.
- Include the "includeSubDomains" to maximize HSTS coverage, and ensure HSTS is enforced on all sub-domains under the current domain
  - Note that this may prevent secure browser access to any sub-domains that utilize HTTP; however, use of HTTP is very severe and highly discouraged, even for websites that do not contain any sensitive information, as their contents can still be tampered via Man-in-the-Middle attacks to phish users under the HTTP domain.
- Once HSTS has been enforced, submit the web-application's address to an HSTS preload list - this will ensure that, even if a client is accessing the web-application for the first time (implying HSTS has not yet been set by the web-application), a browser that respects the HSTS preload list would still treat the web-application as if it had already issued an HSTS header. Note that this requires the server to have a trusted SSL certificate, and issue an HSTS header with a maxAge of 1 year (31536000).
- Note that this query is designed to return one result per application. This means that if more than one vulnerable response without an HSTS header is identified, only the first identified instance of this issue will be highlighted as a result. If a misconfigured instance of HSTS is identified (has a short lifespan, or is missing the "includeSubDomains" flag), that result will be flagged. Since HSTS is required to be enforced across the entire application to consider a secure deployment of HSTS functionality, fixing this issue only where the query highlights this result is likely to produce subsequent results in other sections of the application; therefore, when adding this header via code, ensure it is uniformly deployed across the entire application. If this header is added via configuration, ensure that this configuration applies to the entire application.
- Note that misconfigured HSTS headers that do not contain the recommended max-age value of at least one year or the "includeSubDomains" flag will still return a result for a missing HSTS header.

**Source Code Examples**

**PHP**

Setting the HSTS Header via Code in PHP

```
header("Strict-Transport-Security: max-age=31536000; includeSubDomains");
```

for more information: <http://cve.mitre.org/data/definitions/346.html>

**Query Path:** PHP\cx\PHP\_Medium\_Threat\Missing\_HSTS\_Header Version:1

Hình 73. Missing\_HSTS\_Header: Lỗi hỏng Missing HSTS Header

Query Path: PHP\cx\PHP\_Medium\_Threat\Missing\_HSTS\_Header Version:1

```
1 bool show_only_one_HSTS_result = true;
2 CxList stringLiterals = File.Strings();
3 CxList globalConfig = Check_HSTS_Configuration_In_Code();
4 if(globalConfig.Count > 0){
5     result = Check_HSTS_Configuration();
6 } else{
7     CxList headerChanges = Find_Methods().FindByShortNames(new List<string>(){"header", "setHeader"});
8     CxList parentHeaders = stringLiterals.GetParameters(headerChanges);
9     CxList childHeaders = parentHeaders.FindByShortName("Strict-Transport-Security", false);
10    if(childHeaders.Count == 0){
11        CSharpPage res = All.GetFirstGraph();
12        if(res != null)result.Add(res.NodeId, res);
13    }
14 }
```

Hình 74. Missing\_HSTS\_Header: Lỗi hỏng Missing HSTS Header

## 1. Rủi ro (Risk)

- Nếu không thiết lập HTTP Strict Transport Security (HSTS), trang web có thể bị tấn công Man-in-the-Middle (MITM):
  - Người dùng nhập domain mà không có "https://", trình duyệt mặc định dùng HTTP trước khi chuyển hướng sang HTTPS.
  - Kẻ tấn công có thể chặn yêu cầu HTTP đầu tiên, chuyển hướng đến một trang giả mạo, thu thập thông tin đăng nhập hoặc cài mã độc.
  - Không có HSTS, người dùng có thể bị lừa chấp nhận chứng chỉ SSL không hợp lệ, tạo cơ hội cho MITM.

## 2. Nguyên nhân (Cause)

- Người dùng nhập domain mà không có HTTPS, trình duyệt mặc định dùng HTTP (<http://example.com>).
- Không có HSTS, trình duyệt vẫn chấp nhận HTTP, dễ bị tấn công.

- Không gửi HSTS header từ server, khiến trình duyệt không biết phải luôn sử dụng HTTPS.

### **3. Khuyến nghị chung (General Recommendations)**

- Thiết lập HSTS header trên máy chủ hoặc trong ứng dụng web.
- Cấu hình max-age=31536000 (1 năm) để trình duyệt ghi nhớ HSTS.
- Bật includeSubDomains để bảo vệ toàn bộ subdomain.
- Đăng ký website vào HSTS Preload List để đảm bảo HTTPS ngay cả khi người dùng chưa từng truy cập.

### **4. Ví dụ mã nguồn (Source Code Examples)**

- Thiết lập HSTS trong PHP

```
header("Strict-Transport-Security: max-age=31536000; includeSubDomains");
```

*Hình 75. Thiết lập HSTS trong PHP*

### **5. Kết luận**

- Không có HSTS → Dễ bị tấn công MITM qua HTTP.
- Cách phòng tránh:
  - Bật HSTS với max-age = 1 năm (31536000 giây).
  - Bật includeSubDomains để bảo vệ toàn bộ subdomain.
  - Đăng ký vào HSTS Preload List để bảo vệ toàn diện.
  - Cấu hình HSTS trên server (Apache, Nginx) hoặc trong code (PHP).

## ❖ Path\_Traversal: Lỗi hỏng Path Traversal

The screenshot shows a web browser window with multiple tabs open. The active tab displays information about the 'Path\_Traversal' vulnerability. The page includes sections for 'Risk', 'Cause', 'General Recommendations', and 'Source Code Examples'. A sidebar on the right contains a 'CONFUSED ABOUT PATH TRAVERSAL' section with a link to learn more.

**Risk**  
What might happen  
An attacker could define arbitrary file path for the application to use, potentially leading to:

- Stealing sensitive files, such as configuration or system files
- Overwriting files such as program binaries, configuration files, or system files
- Deleting critical files, causing denial of service (DoS).

In addition to common path traversal issues - allowing attackers to provide absolute paths in PHP is often more severe, due to PHP protocol wrappers and how they work. Protocol wrappers allow various protocols to be acted on using the same functions (e.g. file\_get\_contents, file\_exists) using various schemes (e.g. file:/// , phar:/// , http://). This will often lead to unexpected behavior, depending on implementation and PHP configuration. For example, if an attacker can control a value passed to a file path call they may be able to:

- Perform remote code execution (RCE) with phar:// deserialization attacks
- Perform server-side request forgery (SSRF) with http://
- Filter bypass with php://filter and data://

**Cause**  
How does it happen  
The application uses user input in the file path for accessing files on the application server's local disk.

**General Recommendations**  
How to avoid it

Hình 76. Path\_Traversal: Lỗi hỏng Path Traversal

The screenshot shows a web browser window displaying 'General Recommendations' for avoiding Path\_Traversal. It lists 6 steps:

- Ideally, avoid depending on dynamic data for file selection.
- Validate all input, regardless of source. Validation should be based on a whitelist: accept only data fitting a specified structure, rather than reject bad patterns. Check for:
  - Data type
  - Size
  - Range
  - Format
  - Expected values
- Accept dynamic data only for the filename, not for the path and folders.
- Ensure that file path is fully canonicalized.
- Explicitly limit the application to use a designated folder that is separate from the applications binary folder.
- Restrict the privileges of the application's OS user to necessary files and folders. The application should not be able to write to the application binary folder, and should not read anything outside of the application folder and data folder.

**Source Code Examples**  
PHP  
Absolute Path Traversal in 'filename' Parameter

```
if (isset($_GET['filename'])) {
    $filename = $_GET['filename'];
    if (!is_readable($filename)) {
        $fp = fopen($filename, 'rb');
        fpreadin($fp);
    } else {
        // return 404
    }
} else {
    // return 404
}
```

Hình 77. Path\_Traversal: Lỗi hỏng Path Traversal

```

Source Code Examples
PHP
Absolute Path Traversal in "filename" Parameter

if (isset($_GET['filename'])) {
    $filename = $_GET['filename'];
    if ($is_readable($filename)) {
        $fp = fopen($filename, 'rb');
        fpassthru($fp);
    }
    else {
        // return 404
    }
} else {
    // return 404
}

Relative Path Traversal in "filename" Parameter

if (isset($_GET['filename'])) {
    $filename = $_GET['filename'];
    if ($is_readable($filename)) {
        $fp = fopen($filename, 'rb');
        fpassthru($fp);
    }
    else {
        // return 404
    }
} else {
    // return 404
}

Absolute Path Traversal in "filename" Parameter Leading to Possible SSRF

<?php
/*
 * Filename supplied as an HTTP GET request parameter
 */
if(isset($_GET["filename"])){
    $content = file_get_contents($_GET["filename"]); // Possible sink as the user can supply a filename with "http://", causing SSRF
    // display $content
}
?>

```

Hình 78. Path\_Traversal: Lô hổng Path Traversal

```

Absolute Path Traversal in "filename" Parameter Leading to Possible SSRF

<?php
/*
 * Filename supplied as an HTTP GET request parameter
 */
if(isset($_GET["filename"])){
    $content = file_get_contents($_GET["filename"]); // Possible sink as the user can supply a filename with "http://", causing SSRF
    // display $content
}
?>

Path Traversal Mitigated by Utilizing Basename

if (isset($_GET['filename'])) {
    $filename = "public_file/" . basename($_GET['filename']);
    if ($is_readable($filename)) {
        $fp = fopen($filename, 'rb');
        fpassthru($fp);
    }
    else {
        // return 404
    }
} else {
    // return 404
}

for more information http://cwe.mitre.org/data/definitions/22.html

```

Hình 79. Path\_Traversal: Lô hổng Path Traversal

## 1. Rủi ro (Risk)

- Nếu ứng dụng cho phép kẻ tấn công chỉ định đường dẫn tệp tùy ý, hậu quả có thể bao gồm:
  - Đánh cắp tệp nhạy cảm
    - Truy cập và đọc các tệp quan trọng như:
      - Tệp cấu hình (config.php, .env)

- Tệp hệ thống (/etc/passwd, /etc/shadow)
- Ghi đè hoặc chỉnh sửa tệp quan trọng
  - Thay đổi nội dung tệp mã nguồn hoặc cấu hình của ứng dụng.
  - Ghi đè tệp thực thi (.php, .sh, .exe), có thể dẫn đến tấn công thực thi mã từ xa (RCE).
- Xóa tệp quan trọng, gây gián đoạn dịch vụ (DoS)
  - Xóa các tệp cần thiết khiến ứng dụng bị lỗi hoặc không hoạt động.
- ❖ **Mối nguy hiểm đặc biệt từ PHP Protocol Wrappers**
  - Ngoài Path Traversal thông thường, PHP cho phép sử dụng Protocol Wrappers, giúp thao tác với nhiều giao thức bằng cách hàm như file\_get\_contents(), file\_exists(). Nếu không kiểm soát đầu vào, kẻ tấn công có thể khai thác:
- Thực thi mã từ xa (RCE) với phar://
  - Nếu ứng dụng xử lý tệp .phar, kẻ tấn công có thể tải lên một tệp .phar độc hại và sử dụng deserialization attack để thực thi mã PHP.
- Tấn công Server-Side Request Forgery (SSRF) với http://
  - Nếu ứng dụng hỗ trợ mở tệp từ URL (file\_get\_contents('http://evil.com')), kẻ tấn công có thể buộc ứng dụng gửi request tới nội bộ hệ thống (<http://localhost/admin>).
- Bypass kiểm tra nội dung với php://filter hoặc data://
  - php://filter có thể được sử dụng để đọc mã nguồn của tệp PHP mà không thực thi nó.
  - data:// có thể cho phép kẻ tấn công nhúng mã độc trực tiếp trong yêu cầu.

## 2. Nguyên nhân (Cause)

- Ứng dụng dùng input của người dùng để xác định file trên server.
- Không kiểm tra hoặc kiểm tra không đầy đủ các ký tự đặc biệt (., /, \).
- Cho phép sử dụng đường dẫn tuyệt đối hoặc protocol wrappers (file://, http://).

## 3. Khuyến nghị chung (General Recommendations)

- Tránh phụ thuộc vào dữ liệu động để chọn tệp
  - Hạn chế việc sử dụng đầu vào từ người dùng để xác định đường dẫn tệp.
- Xác thực tất cả đầu vào, bắt kể nguồn gốc
  - Sử dụng danh sách trắng (whitelist) thay vì danh sách đen (blacklist).
  - Kiểm tra các tiêu chí sau đối với đầu vào:

- Kiểu dữ liệu (Data type)
  - Kích thước (Size)
  - Phạm vi giá trị hợp lệ (Range)
  - Định dạng (Format)
  - Giá trị mong đợi (Expected values)
- Chỉ cho phép dữ liệu động cho tên tệp, không cho phép đường dẫn thư mục
    - Tránh để người dùng nhập đường dẫn đầy đủ.
    - Sử dụng các biện pháp kiểm tra để đảm bảo chỉ có tên tệp được cung cấp.
  - Đảm bảo đường dẫn tệp đã được chuẩn hóa hoàn toàn (canonicalized)
    - Dùng các hàm như realpath() để xử lý đường dẫn và loại bỏ các ký tự bất thường như ../ hoặc ../../
  - Hạn chế ứng dụng chỉ sử dụng một thư mục được chỉ định riêng biệt
    - Thiết lập một thư mục cố định để lưu và truy xuất tệp.
    - Không cho phép truy cập đến các thư mục quan trọng khác trong hệ thống.
  - Hạn chế quyền truy cập của người dùng hệ điều hành (OS user privileges)
    - Chỉ cấp quyền tối thiểu cần thiết để ứng dụng hoạt động.
    - Ứng dụng không nên có quyền ghi vào thư mục chứa mã nguồn (binary folder).
    - Ứng dụng không nên có quyền đọc bất kỳ tệp nào ngoài thư mục dữ liệu hoặc thư mục được chỉ định.

#### 4. Ví dụ mã nguồn (Source Code Examples)

- Tấn công Path Traversal với đường dẫn tuyệt đối trong tham số "filename"

```

if (isset($_GET['filename'])) {
    $filename = $_GET['filename'];
    if (is_readable($filename)) {
        $fp = fopen($filename, 'rb');
        fpassthru ($fp);
    }
    else {
        // return 404
    }
} else {
    // return 404
}

```

Hình 80. Tấn công Path Traversal với đường dẫn tuyệt đối trong tham số "filename"

- Tấn công Path Traversal với đường dẫn tương đối trong tham số "filename"

```

if (isset($_GET['filename'])) {
    $filename = "public_files/".$_GET['filename'];
    if (is_readable($filename)) {
        $fp = fopen($filename, 'rb');
        fpassthru ($fp);
    }
    else {
        // return 404
    }
} else {
    // return 404
}

```

*Hình 81. Tấn công Path Traversal với đường dẫn tương đối trong tham số "filename"*

- Tấn công Path Traversal với đường dẫn tuyệt đối trong tham số "filename", có thể dẫn đến tấn công SSRF

```

<?php
/**
 * Filename supplied as an HTTP GET request parameter
*/
if(isset($_GET["filename"])) {
    $content = file_get_contents($_GET["filename"]); // Possible sink as the user can supply a filename with "http://", causing SSRF
    // display $content
}
?>

```

*Hình 82. - Tấn công Path Traversal với đường dẫn tuyệt đối trong tham số "filename", có thể dẫn đến tấn công SSRF*

- Giảm thiểu tấn công Path Traversal bằng cách sử dụng basename

```

if (isset($_GET['filename'])) {
    $filename = "public_files/".basename($_GET['filename']);
    if (is_readable($filename)) {
        $fp = fopen($filename, 'rb');
        fpassthru ($fp);
    }
    else {
        // return 404
    }
} else {
    // return 404
}

```

*Hình 83. Giảm thiểu tấn công Path Traversal bằng cách sử dụng basename*

## 5. Kết luận

- Lỗ hổng Path Traversal rất nguy hiểm, có thể dẫn đến rò rỉ dữ liệu, thực thi mã từ xa, hoặc tấn công SSRF.

- Cách phòng tránh hiệu quả:

- Chỉ cho phép tên file, không cho phép đường dẫn.
- Dùng basename() để loại bỏ ký tự ../../
- Giới hạn truy cập vào một thư mục cố định.
- Không cấp quyền ghi vào thư mục quan trọng.

❖ **Possible\_Flow\_Control:** Kiểm Soát Luồng Chương Trình Bị Ánh Hưởng

The screenshot shows a web browser window with multiple tabs. The active tab displays information about the 'Possible\_Flow\_Control' vulnerability. The content includes:

- Risk**: What might happen: An attacker with the ability of controlling program's flow might control the output of the program and result with an unexpected output.
- Cause**: How does it happen: This will occur when variables from user input are used in control flow decisions.
- General Recommendations**: How to avoid it: User input should not be taken into consideration as part of program's control flow.
- Source Code Examples**: Java code example:

```
boolean public login(user) {
    while(user){
        //do something
    }
    return true;
}
```
- User input does not affects the while loop**

Hình 84. Possible\_Flow\_Control: Kiểm Soát Luồng Chương Trình Bị Ánh Hưởng

```

Source Code Examples
Java
User input affects the while loop

boolean public login(user) {
    while(user) {
        // do something
    }
    return true;
}

User input does not affect the while loop

boolean isAdmin = false;
boolean public login(user) {
    boolean isAdmin = isAuthorized(user);
    while(isAdmin) {
        // do something
    }
    return true;
}

for more information http://cwe.mitre.org/data/definitions/691.html

Query Path: PHP/CxPHP_Low_Visibility/Possible_Flow_Control Version:1
[Query Source]
1 CxList Inputs = Find_Inputs();
2 Inputs.Add(Find_Read());
3 Inputs.Add(Find_File_Read());
4 CxList sanitized = Find_Flow_Control_Sanitize();
5 CxList flowClauses = All.NewCxList();
6 try
7 {
8     foreach( Cxlist statement in All.FindByType(typeof(SwitchStmt)) )
9     {
10         SwitchStmt g = statement.TryGetSharpGraph<SwitchStmt>();
11         flowClauses.Add(All.FindById(g.Condition.NodeId));
12     }
13 }
14

```

*Hình 85. Possible\_Flow\_Control: Kiểm Soát Luồng Chương Trình Bị Ảnh Hưởng*

### 1. Rủi ro (Risk)

- Khi kẻ tấn công có thể kiểm soát luồng thực thi của chương trình, chúng có thể:
  - Gây ra kết quả không mong muốn.
  - Thay đổi hành vi ứng dụng, có thể dẫn đến lỗ hổng bảo mật nghiêm trọng.

### 2. Nguyên nhân (Cause)

- Lỗi này xảy ra khi biến đầu vào từ người dùng được sử dụng để quyết định luồng điều khiển của chương trình.
- Ví dụ:
  - Nếu một biến nhập từ người dùng (userInput) được dùng để điều khiển vòng lặp (while, for) hoặc câu lệnh điều kiện (if), kẻ tấn công có thể thay đổi logic của chương trình theo ý muốn.

### 3. Khuyến nghị chung (General Recommendations)

- Không sử dụng trực tiếp đầu vào từ người dùng để quyết định luồng chương trình.
- Tách biệt luồng điều khiển bằng cách xác thực quyền truy cập trước khi thực hiện hành động.

- Sử dụng các giá trị đã được kiểm tra hoặc tính toán thay vì dùng trực tiếp đầu vào từ người dùng

#### **4. Ví dụ mã nguồn (Source Code Examples)**

- Mã dễ bị tấn công (User input ảnh hưởng vòng lặp while)

```
boolean public login(user) {
    while(user) {
        //do something
    }
    return true;
}
```

*Hình 86. Mã dễ bị tấn công (User input ảnh hưởng vòng lặp while)*

- Rủi ro: Kẻ tấn công có thể cung cấp giá trị user để kiểm soát vòng lặp, làm chương trình bị treo hoặc thực hiện hành vi không mong muốn.
- Cách khắc phục (Luồng điều khiển tách biệt với user input)

```
boolean isAdmin = false;

boolean public login(user) {
    boolean isAdmin = isAuthorized(user);
    while(isAdmin) {
        // do something
    }
    return true;
}
```

*Hình 87. Cách khắc phục (Luồng điều khiển tách biệt với user input)*

- Lợi ích: Chỉ khi user đã được xác thực hợp lệ, vòng lặp mới chạy.

#### **5. Kết luận**

- Không sử dụng trực tiếp dữ liệu từ người dùng để kiểm soát luồng chương trình.
  - Xác thực dữ liệu đầu vào trước khi đưa vào cấu trúc điều khiển (if, while, for).
  - Tách biệt luồng điều khiển khỏi đầu vào của người dùng, chỉ sử dụng các giá trị đã được xác minh.



## CHƯƠNG IV: ĐÁNH GIÁ

Sau quá trình nghiên cứu và thực hiện kiểm thử xâm nhập (Pentest), đề tài đã đạt được những kết quả quan trọng trong việc xác định các lỗ hổng bảo mật của hệ thống. Dưới đây là một số đánh giá chính:

Tính hiệu quả của phương pháp: Các công cụ và kỹ thuật được sử dụng trong quá trình kiểm thử xâm nhập đã giúp phát hiện và đánh giá chính xác các điểm yếu trong hệ thống.

Khả năng khai thác lỗ hổng: Một số lỗ hổng quan trọng đã được phát hiện, bao gồm [liệt kê một số lỗ hổng tiêu biểu như SQL Injection, XSS, v.v. nếu có]. Điều này cho thấy hệ thống cần có các biện pháp khắc phục kịp thời để đảm bảo an toàn.

Tính khả thi của các giải pháp bảo mật: Dựa trên các lỗ hổng được tìm thấy, các giải pháp bảo mật đã được đề xuất nhằm giảm thiểu rủi ro, giúp hệ thống trở nên an toàn hơn trước các cuộc tấn công tiềm năng.

Hạn chế của nghiên cứu: Một số phương pháp kiểm thử có thể chưa được triển khai đầy đủ do hạn chế về thời gian, công cụ hoặc quyền truy cập hệ thống.

## CHƯƠNG V: KẾT LUẬN

- Từ các kết quả kiểm thử, có thể rút ra một số kết luận quan trọng như sau:
  - Hệ thống hiện tại vẫn tồn tại những lỗ hổng bảo mật có thể bị khai thác, đòi hỏi sự quan tâm và biện pháp khắc phục kịp thời.
  - Việc áp dụng kiểm thử xâm nhập định kỳ là cần thiết để duy trì mức độ an toàn của hệ thống.
  - Các biện pháp bảo mật như cập nhật phần mềm, kiểm soát truy cập chặt chẽ, mã hóa dữ liệu và đào tạo nhân sự về an toàn thông tin là những yếu tố quan trọng cần được triển khai.
  - Trong tương lai, nghiên cứu có thể mở rộng bằng cách áp dụng các kỹ thuật kiểm thử nâng cao hơn, sử dụng trí tuệ nhân tạo hoặc machine learning để tự động phát hiện và khắc phục lỗ hổng.

Tóm lại, nghiên cứu này đã cung cấp một cái nhìn tổng quan về tình trạng bảo mật của hệ thống, đồng thời đề xuất các biện pháp cải thiện nhằm đảm bảo an toàn thông tin. Việc tiếp tục nghiên cứu và ứng dụng các phương pháp bảo mật tiên tiến là điều cần thiết để đối phó với các mối đe dọa ngày càng tinh vi trong không gian mạng.

## TÀI LIỆU THAM KHẢO

1. <https://cystack.net/vi/blog/pentest-la-gi#quy-trinh-pentest>
2. [https://vi.wikipedia.org/wiki/Ki%E1%BB%83m\\_tra\\_th%C3%A2m\\_nh%E1%BA%ADp](https://vi.wikipedia.org/wiki/Ki%E1%BB%83m_tra_th%C3%A2m_nh%E1%BA%ADp)
3. [https://tenten.vn/tin-tuc/pentest-la-gi/#Caacuteec\\_b4327899c\\_th7921c\\_hi7879n\\_Pentest](https://tenten.vn/tin-tuc/pentest-la-gi/#Caacuteec_b4327899c_th7921c_hi7879n_Pentest)