Bài tập buổi 2

Họ tên	MSSV	Phân chia công việc
Trần Đức Trí Dũng	21520748	Câu 1: 3 nguyên tắc cuối + câu 2
Trần Thanh Triều	21522713	Câu 1: 5 nguyên tắc tiếp theo
Nguyễn Đình Luân	21521105	Câu 1: 5 nguyên tắc đầu tiên

Câu 1: Tóm tắt tất cả các Nguyên tắc Thiết kế Bảo mật (Security DesignPrinciples) và đưa ra ví dụ cho từng nguyên tắc.

Economy of Mechanism (đơn giản trong sử dụng) là một nguyên tắc thiết kế quan trọng đề cập đến việc giảm bớt sự phức tạp trong hệ thống bảo mật và giữ cho cơ chế bảo mật đơn giản và hiệu quả. Nguyên tắc này có ý định làm cho hệ thống bảo mật dễ hiểu, dễ quản lý và dễ kiểm tra.

Ví dụ: Hệ thống xác thực đơn giản:

Một hệ thống xác thực dựa trên nguyên tắc "Economy of Mechanism" sẽ thiết kế quá trình xác thực một cách đơn giản và hiệu quả. Thay vì có nhiều bước xác thực phức tạp, hệ thống có thể yêu cầu chỉ một tên người dùng và mật khẩu. Điều này làm giảm sự phức tạp trong việc quản lý và duy trì hệ thống xác thực.

Fail-Safe Defaults (Mặc định an toàn) đề cập đến việc thiết lập các giá trị và cài đặt mặc định sao cho hệ thống hoặc ứng dụng hoạt động một cách an toàn khi không có cài đặt cụ thể hoặc khi có lỗi. Mục tiêu của nguyên tắc này là bảo vệ hệ thống hoặc dữ liệu khỏi các trạng thái không an toàn.

Ví dụ: trong một ứng dụng di động, nếu người dùng không chọn cài đặt quyền truy cập riêng tư cho ảnh mới chụp, ứng dụng tự động đặt cài đặt mặc định là "ảnh riêng tư" để đảm bảo tính riêng tư của người dùng.

Open Design (Thiết kế Mở) đề cập đến việc công khai hoặc chia sẻ thông tin về thiết kế và hoạt động của hệ thống hoặc ứng dụng bảo mật. Mục tiêu của nguyên tắc này là tạo sự minh bạch và khuyến khích đánh giá từ cộng đồng để cải thiện tính bảo mật.

Ví dụ: Trình duyệt mã nguồn mở: Mozilla Firefox là một trình duyệt web mã nguồn mở. Mã nguồn của Firefox có sẵn công khai và có một cộng đồng lớn của nhà phát triển và chuyên gia bảo mật thường xuyên kiểm tra mã nguồn và báo cáo về các lỗ hổng bảo mật.

Complete Mediation (Trung gian hoàn toàn): đề cập đến việc mọi truy cập vào tài nguyên hoặc chức năng phải được kiểm tra và quyết định về quyền truy cập của người dùng một cách hoàn toàn trước khi truy cập đó được chấp nhận. Nguyên tắc này đảm bảo rằng không có truy cập nào được thực hiện mà không được kiểm tra và phê duyệt.

Ví dụ: Trong hệ thống quản lý tệp, complete mediation đảm bảo rằng mọi yêu cầu truy cập tới một tệp cụ thể phải được kiểm tra và phê duyệt trước khi người dùng được phép xem hoặc chỉnh sửa tệp đó. Chẳng hạn, nếu người dùng muốn xem một tệp, hệ thống sẽ kiểm tra xem họ có quyền truy cập tệp đó hay không trước khi cho phép họ xem nội dung. Điều này đảm bảo tính toàn vẹn của quản lý quyền truy cập và ngăn chặn việc truy cập trái phép đến dữ liệu nhạy cảm.

Least privilege: nguyên tắc thiết kế bảo mật Least privilege là một phương pháp tốt trong an toàn thông tin giúp các tổ chức giảm thiểu rủi ro bằng cách chỉ cấp cho cá nhân hoặc thực thể các quyền và đặc quyền truy cập cần thiết để thực hiện nhiệm vụ của mình. Phương pháp này giúp giảm thiểu các lỗ hổng bảo mật tiềm năng và giới hạn tác động tiềm năng của các cuộc tấn công bảo mật.

Một ví dụ về áp dụng nguyên tắc thiết kế bảo mật Least privilege trong một tập đoàn là việc chỉ cấp quyền quản trị viên hệ thống cho nhóm nhân viên IT có trách nhiệm quản lý và bảo mật hệ thống. Các nhóm nhân viên khác, chẳng hạn như kế toán hoặc nhân sự, chỉ được cấp quyền truy cập vào các ứng dụng và dữ liệu liên quan đến công việc của họ. Nguyên tắc least privilege đảm bảo rằng chỉ những người cần thiết mới có thể truy cập và quản lý toàn bộ hệ thống, giảm thiểu khả năng lợi dụng hoặc tiềm năng rủi ro từ những tài khoản không cần thiết.

Least common mechanism: nguyên tắc Thiết kế Bảo mật Least Common Mechanism tập trung vào việc giảm thiểu sự chia sẻ cơ chế giữa các thành phần trong hệ thống, nhằm giảm thiểu nguy cơ ảnh hưởng và lợi dụng từ các lỗ hổng bảo mật. Áp dụng nguyên tắc này giúp hạn chế sự tác động tiêu cực từ một lỗ hổng chỉ ảnh hưởng đến một phần hạn chế của hệ thống, giữ cho toàn bộ hệ thống an toàn và bảo mật hơn.

Một ví dụ về việc áp dụng nguyên tắc LCM trong một hệ thống là cấp phát các quyền truy cập và cơ chế cho các tài khoản người dùng khác nhau. Thay vì cho phép tất cả người dùng có cùng quyền truy cập vào tất cả các thành phần của hệ thống, nguyên tắc LCM yêu cầu gán quyền truy cập dựa trên nhiệm vụ và trách nhiệm cụ thể của mỗi người dùng. Ví dụ, người dùng văn phòng chỉ có quyền truy cập vào ứng dụng văn phòng và tài liệu liên quan, trong khi nhóm phát triển chỉ có quyền truy cập vào các công cụ và môi trường phát triển. Điều này giảm thiểu khả năng một lỗ hổng trong một thành phần có thể tác động đến các thành phần khác trong hệ thống.

Psychological acceptability: Nguyên tắc Thiết kế Bảo mật Psychological Acceptability tập trung vào việc thiết kế các biện pháp bảo mật sao cho người dùng có thể tiếp nhận và sử dụng một cách thuận tiện, mà không gây cản trở đáng kể đến hiệu suất hoặc trải nghiệm của họ. Áp dụng nguyên tắc này giúp tăng cường sự tuân thủ và chấp nhận từ phía người dùng, đồng thời nâng cao mức độ bảo mật và giảm thiểu nguy cơ từ việc bỏ qua các biện pháp bảo mật.

Một ví dụ về việc áp dụng nguyên tắc Psychological Acceptability trong một hệ thống là việc thiết kế chính sách đặt mật khẩu hợp lý và dễ nhớ. Thay vì yêu cầu người dùng tạo mật khẩu phức tạp, chẳng hạn gồm ký tự đặc biệt, chữ hoa, chữ thường và số, mà nhiều người dùng có thể dễ dàng quên, nguyên tắc này khuyến nghị sử dụng mật khẩu dễ nhớ, nhưng vẫn đủ mạnh và không dễ bị đoán đến. Ví dụ, thay vì "P@ssw0rd!" có thể yêu cầu người dùng tạo mật khẩu như "GardenCat27!". Điều này giúp tăng cường sự chấp nhận và tuân thủ từ phía người dùng, đồng thời giảm thiểu khả năng họ sẽ vi phạm bảo mật bằng cách sử dụng mật khẩu dễ đoán hoặc lưu giữ trái phép.

Isolation: nguyên tắc Thiết kế Bảo mật Isolation tập trung vào việc cách ly các thành phần khác nhau của hệ thống nhằm giới hạn sự tương tác và ảnh hưởng giữa chúng. Áp dụng nguyên tắc này giúp giảm thiểu khả năng lây lan lỗ hổng và tác động của cuộc tấn công của một thành phần đến các thành phần khác, đảm bảo tính bảo mật toàn diện cho hệ thống.

Một ví dụ về việc áp dụng nguyên tắc Isolation trong một hệ thống là sử dụng mô hình cách ly mạng. Thay vì cho phép tất cả các phần trong mạng nội bộ của một công ty có khả năng truy cập vào các thông tin nhạy cảm, nguyên tắc này yêu cầu phân chia các mạng con riêng biệt để cách ly các phần khác nhau của công ty. Ví dụ, có thể có mạng nội bộ cho các văn phòng và mạng riêng biệt cho các trung tâm dữ liệu hoặc mạng Wi-Fi khách. Cách ly mạng giảm thiểu khả năng một cuộc tấn công từ bên ngoài xâm nhập và lan truyền sang các mạng khác.

Encapsulation: nguyên tắc Thiết kế Bảo mật Encapsulation tập trung vào việc bảo vệ thông tin và quyền truy cập bằng cách đóng gói chúng vào các phần tử độc lập và cung cấp quyền truy cập thông qua các giao diện kiểm soát. Áp dụng nguyên tắc này giúp giữ cho thông tin nhạy cảm và chức năng của một thành phần hoặc người dùng riêng tư, chỉ tiết lộ thông tin và chức năng cần thiết qua các giao diện xác định.

Một ví dụ về việc áp dụng nguyên tắc Encapsulation trong một hệ thống là thiết kế của một ứng dụng trực tuyến chia sẻ file. Trong ứng dụng này, thông tin và chức năng của mỗi người dùng đóng gói vào tài khoản riêng tư của họ. Người dùng chỉ có thể truy cập và quản lý tập tin của mình thông qua giao diện đã được thiết lập và kiểm soát. Điều này giúp đảm bảo rằng người dùng không thể truy cập hoặc chỉnh sửa thông tin hoặc tài liệu của người dùng khác thông qua ứng dụng.

Modularitzy: là việc phát triển các chức năng bảo mật thành các module riêng biệt. Các chức năng bảo mật như mã hóa được phát triển thành các module chung để giúp nhiều giao thức và ứng dụng có thể sử dụng chúng mà không cần triển khai lại từng cái. Điều này cũng hỗ trợ chuyển đổi công nghệ mới hoặc nâng cấp tính năng mà không cần thay đổi toàn bộ hệ thống

Ví dụ: tạo một module mã hóa chung để dùng cho nhiều ứng dụng hoặc giao thức. Các ứng dụng email, lưu trữ tệp, ứng dụng trò chuyện,… có thể sử dụng cùng

module mã hóa để bảo vệ dữ liệu. Khi cần thì chỉ cập nhật module này chứ không phải thay đổi từng ứng dụng

Layering: sử dụng nhiều phương pháp chồng chéo để giải quyết khía cạnh con người, công nghệ và vận hành của hệ thống. Nếu có một phương pháp bảo mật bị phá vỡ vẫn sẽ có những lớp bảo mật khác đảm bảo sự an toàn cho hệ thống

Ví dụ: sử dụng tường lửa để ngăn chặn truy cập trái phép, hệ thống phát hiện xâm nhập để theo dõi và phát hiện các cuộc tấn công, quyền truy cập người dùng và sao lưu dữ liệu để có thể khôi phục dữ liệu phòng khi bị tấn công.

Least astonishment: là nguyên tắc thiết kế trong giao diện người dùng, nó đề cập đến việc một chương trình hoặc giao diện người dùng phải phản ứng theo cách ít gây ngạc nhiên nhất cho người dùng.

Ví dụ: khi người dùng muốn truy cập vào một trang web, hệ thống cần cung cấp giao diện để người dùng có thể dễ dàng hiểu và tuân thủ. Nếu hệ thống không tuân theo nguyên tắc này, người dùng có thể bối rối bởi các yêu cầu xác thực phức tạp hoặc thông báo không rõ ràng, dẫn đến họ không hiểu các hoạt động của hệ thống và dễ dàng gặp lỗi bảo mật

Câu 2:

Trong chương trình trên, khi quyền truy cập bị từ chối (dwRet = “ERROR_ACCESS_DENIED”) chương trình chỉ có dòng ghi chú “Inform user that access is denied” mà không có biện pháp xử lí cụ thể để đảm bảo tính an toàn cũng như làm người dùng bối rối do không cung cấp thông báo cũng như hướng khắc phục.

Viết lại đoạn mã

```
DWORD dwRet = IsAccessAllowed(...);
if (dwRet == ERROR_ACCESS_DENIED)
{
    DisplayAccessDeniedMessage();
}
else if (dwRet == ERROR_SUCCESS)
{
    DoSomething();
}
else
{
    HandleOtherErrors(dwRet);
}

void DisplayAccessDeniedMessage()
{
    MessageBox(NULL, L"Access Denied. Please contact support for assistance.", L"Access Denied", MB_ICONERROR);
}

void DoSomething() ···

void HandleOtherErrors(DWORD dwError) ···
```

Giải thích:

- Khi quyền truy cập truy cập bị từ chối, ta dùng hàm “DisplayAccessDeniedMessage” để thông báo cho người dùng và cung cấp hướng dẫn để giải quyết vấn đề.

- Khi quyền truy cập được chấp nhận, ta thực hiện công việc cần thiết với hàm “Dosomething”

- Nếu xảy ra lỗi khác, ta dùng hàm “HandleOtherError” để xử lý