TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

**LỚP CỬ NHÂN TÀI NĂNG**

ĐỖ TRUNG ĐỨC – TRẦN QUỐC TÂM

GIẢI PHÁP PHẦN MỀM

BẢO VỆ CHỐNG THẤT THOÁT DỮ LIỆU

TRÊN MÁY TÍNH CÁ NHÂN

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN CNTT

TP.HCM, 2015

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

**LỚP CỬ NHÂN TÀI NĂNG**

ĐỖ TRUNG ĐỨC 1112075

TRẦN QUỐC TÂM 1112276

GIẢI PHÁP PHẦN MỀM

BẢO VỆ CHỐNG THẤT THOÁT DỮ LIỆU

TRÊN MÁY TÍNH CÁ NHÂN

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN TIN HỌC

GIÁO VIÊN HƯỚNG DẪN

TS.TrẦn Minh TriẾt

NIÊN KHÓA 2011– 2015

|  |
| --- |
| **NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  Khóa luận đáp ứng yêu cầu của LV cử nhân tin học.  TpHCM, ngày …… tháng …… năm 2015  Giáo viên hướng dẫn |
| **NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN**  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  Khóa luận đáp ứng yêu cầu của LV cử nhân tin học.  TpHCM, ngày …… tháng …… năm 2015  Giáo viên phản biện |

LỜI CÁM ƠN

Chúng em xin chân thành cảm ơn Khoa Công Nghệ Thông Tin, trường Đại Học Khoa Học Tự Nhiên, Tp.HCM đã tạo điều kiện tốt cho chúng em thực hiện đề tài này.

Chúng em xin chân thành cảm ơn Thầy Trần Minh Triết, là người đã tận tình hướng dẫn, chỉ bảo chúng em trong suốt thời gian thực hiện đề tài. Chúng em cũng xin cảm ơn Thầy Lương Vi Minh đã có những trao đổi, những chỉ dẫn giúp chúng em giải quyết các vấn đề và hoàn thiện đề tài.

Chúng em cũng xin gửi lời cảm ơn sâu sắc đến quý Thầy Cô trong Khoa đã tận tình giảng dạy, trang bị cho chúng em những kiến thức quí báu trong những năm học vừa qua.

Chúng em xin gửi lòng biết ơn sâu sắc đến Ba, Mẹ, các anh chị và bạn bè đã ủng hộ, giúp đỡ và động viên chúng em trong những lúc khó khăn cũng như trong suốt thời gian học tập và nghiên cứu.

Mặc dù chúng em đã cố gắng hoàn thành luận văn trong phạm vi và khả năng cho phép, nhưng chắc chắn sẽ không tránh khỏi những thiếu sót, kính mong sự cảm thông và tận tình chỉ bảo của quý Thầy Cô và các bạn.

Nhóm thực hiện

Đỗ Trung Đức & Trần Quốc Tâm

ĐỀ CƯƠNG CHI TIẾT

|  |
| --- |
| **Tên Đề Tài**: Bảo vệ dữ liệu trên thiết bị lưu trữ di động |
| **Giáo viên hướng dẫn**: TS. Trần Minh Triết |
| **Thời gian thực hiện**: từ ngày 25/02 /2015 đến ngày 15/07 /2015 |
| **Sinh viên thực hiện:**  Đỗ Trung Đức (1112075) – Trần Quôc Tâm (1112276) |
| **Loại đề tài:** Nghiên cứu lý thuyết, giải pháp kỹ thuật và hiện thực hóa sản phẩm phần mềm. |
| **Nội Dung Đề Tài** (mô tả chi tiết nội dung đề tài, yêu cầu, phương pháp thực hiện, kết quả đạt được, …):   * Nghiên cứu tổng quan tình hình thất thoát dữ liệu, đặc biệt là trên các thiết bị lưu trữ di dộng. * Phân tích các phương pháp can thiệp hệ thống, bảo vệ dữ liệu trên máy tính và khi sao chép ra các thiết bị lưu trữ. * Phát triển ứng dụng bảo vệ dữ liệu. |

|  |  |
| --- | --- |
| Kế Hoạch Thực Hiện:  25/11/2014-25/12/2015: Tìm hiểu tình hình thất thoát dữ liệu.  25/12/2014-15/02/2015: Khảo sát các giải pháp chống thất thoát dữ liệu.  15/02/2015-15/03/2015: Tìm hiểu các phương pháp can thiệp hệ thống.  16/03/2015-29/04/2015: Nghiên cứu và phân tích giải pháp bảo vệ dữ liệu.  01/05/2015-20/05/2015: Thiết kế kiến trúc hệ thống cho giải pháp phần mềm.  20/05/2015-30/06/2015: Xây dựng hoàn chỉnh ứng dụng.  01/07/2015-10/07/2015: Tiến hành chạy thử nghiệm. | |
| **Xác nhận của GVHD**  **TS. Trần Minh Triết** | **Ngày tháng năm** 2015  **Nhóm SV Thực hiện**  **Đỗ Trung Đức – Trần Quốc Tâm** |

MỤC LỤC

[LỜI CÁM ƠN 3](#_Toc423818329)

[ĐỀ CƯƠNG CHI TIẾT 4](#_Toc423818330)

[MỤC LỤC 6](#_Toc423818331)

[DANH MỤC CÁC HÌNH 10](#_Toc423818332)

[DANH MỤC CÁC BẢNG 13](#_Toc423818333)

[TÓM TẮT KHÓA LUẬN 14](#_Toc423818334)

[Chương 1 Mở đầu 1](#_Toc423818335)

[1.1. Giới thiệu 1](#_Toc423818336)

[1.2. Lý do thực hiện đề tài 5](#_Toc423818337)

[1.3. Mục tiêu đề tài 6](#_Toc423818338)

[1.4. Nội dung luận văn 6](#_Toc423818339)

[Chương 2 Tổng quan về các biện pháp ngăn ngừa thất thoát dữ liệu 8](#_Toc423818340)

[2.1. McAfee 8](#_Toc423818341)

[2.1.1. Giải pháp phần mềm 8](#_Toc423818342)

[2.1.2. Giải pháp phần cứng (thường có giá thành trên $12000) 9](#_Toc423818343)

[2.1.3. Các tính năng chung của McAfee DLP Endpoint 9](#_Toc423818344)

[2.1.4. Trial McAfee Complete Endpoint Advanced 10](#_Toc423818345)

[2.2. TrendMicro 13](#_Toc423818346)

[2.2.1. Giải pháp phần mềm 13](#_Toc423818347)

[2.2.2. Các tính năng của các sản phẩm tích hợp Trend Micro Intergrated Data Loss Prevention 13](#_Toc423818348)

[2.2.3. Trial TrendMicro 14](#_Toc423818349)

[2.3. Symantec 17](#_Toc423818350)

[2.3.1. Giải pháp phần mềm 17](#_Toc423818351)

[2.3.2. Các tính năng chống mất mát dữ liệu [11] 18](#_Toc423818352)

[2.3.3. Trial Symantec File Share Encryption 21](#_Toc423818353)

[2.4. RSA 24](#_Toc423818354)

[2.4.1. Giải pháp phần mềm 24](#_Toc423818355)

[2.4.2. Các tính năng chống mất mát dữ liệu [12] 24](#_Toc423818356)

[2.5. WebSense 25](#_Toc423818357)

[2.5.1. Giải pháp phần mềm [13] 25](#_Toc423818358)

[2.6. Kết luận 27](#_Toc423818359)

[Chương 3 Can thiệp hệ thống 28](#_Toc423818360)

[3.1. Hook API 28](#_Toc423818361)

[3.2. Kỹ thuật chèn DLL vào process 29](#_Toc423818362)

[3.2.1. Registry 29](#_Toc423818363)

[3.2.2. System-wide Windows Hooks 29](#_Toc423818364)

[3.2.3. Injecting DLL by using CreateRemoteThread() API function 31](#_Toc423818365)

[3.3. Cơ chế chặn hàm API 33](#_Toc423818366)

[3.3.1. Windows subclassing 34](#_Toc423818367)

[3.3.2. Proxy DLL 35](#_Toc423818368)

[3.3.3. Code overwriting 35](#_Toc423818369)

[3.3.4. Spying by altering of the Import Address Table 36](#_Toc423818370)

[3.4. Microsoft Office Add-in 36](#_Toc423818371)

[3.4.1. Sử dụng cơ chế mã hóa mặc định của Word 37](#_Toc423818372)

[3.4.2. Tự mã hóa dữ liệu 43](#_Toc423818373)

[3.5. Phát hiện thiết bị lưu trữ tương tác với máy tính 43](#_Toc423818374)

[3.5.1. Giải pháp C++ 43](#_Toc423818375)

[3.5.2. Giải pháp C# 50](#_Toc423818376)

[Chương 4 Bảo vệ dữ liệu 56](#_Toc423818377)

[4.1. Giải pháp bảo vệ dữ liệu 56](#_Toc423818378)

[4.1.1. Chế độ mã hóa (Mode cipher) 56](#_Toc423818379)

[4.1.2. Mã hóa và giải mã 57](#_Toc423818380)

[4.2. Thông tin mã hóa (metadata) 58](#_Toc423818381)

[4.2.1. Thông tin mã hóa chứa những gì? 58](#_Toc423818382)

[4.2.2. Lưu thông tin mã hóa ngay trong file được mã hóa. 59](#_Toc423818383)

[4.2.3. Lưu thông tin mã hóa ra một file riêng biệt. 63](#_Toc423818384)

[4.3. Kết luận 64](#_Toc423818385)

[Chương 5 Kiến trúc hệ thống 65](#_Toc423818386)

[5.1. Kiến trúc hệ thống 65](#_Toc423818387)

[5.2. Phân hệ can thiệp hệ thống – Hook Module 66](#_Toc423818388)

[5.2.1. DLL Injection Module 67](#_Toc423818389)

[5.2.2. API Intercepting Module 68](#_Toc423818390)

[5.2.3. Hoạt động của ứng dụng. 69](#_Toc423818391)

[5.3. Phân hệ bảo vệ dữ liệu – Protect data 75](#_Toc423818392)

[5.3.1. Tạo Key mã hóa – Key Engine 76](#_Toc423818393)

[5.3.2. Thông tin mã hóa – metadata 76](#_Toc423818394)

[5.3.3. Mã hóa và giải mã – Crypto Engine 77](#_Toc423818395)

[5.4. Phân hệ tương tác với người dùng – User Interface 78](#_Toc423818396)

[5.4.1. Giao diện chính – Explorer 78](#_Toc423818397)

[5.4.2. Chứng thực người dùng – Authentication 78](#_Toc423818398)

[5.4.3. Các loại file hỗ trợ bảo vệ - File Extension 79](#_Toc423818399)

[5.4.4. Ứng dụng chạy nền – System Stray 79](#_Toc423818400)

[Chương 6 Kết luận 80](#_Toc423818401)

[6.1. Các kết quả đạt được 80](#_Toc423818402)

[6.2. Hướng phát triển của đề tài 82](#_Toc423818403)

DANH MỤC CÁC HÌNH

[Hình 1‑1 Số lượng thông tin rò rỉ từ năm 2006-2013 [2] 2](#_Toc423818252)

[Hình 1‑2 Tỷ lệ thông tin rò rỉ vô tình và có chủ ý năm 2012 và 2013 [2] 2](#_Toc423818253)

[Hình 1‑3 Tỷ lệ kiểu thông tin bị thất thoát [2] 3](#_Toc423818254)

[Hình 1‑4 Tỷ lệ rò rỉ dữ liệu theo các kênh [1] 4](#_Toc423818255)

[Hình 1‑5 Tỷ lệ có chủ ý và vô tình rò rỉ dữ liệu theo kênh năm 2013 [2] 5](#_Toc423818256)

[Hình 2‑1 Giao diện đăng nhập 10](#_Toc423818257)

[Hình 2‑2 Dashboard 11](#_Toc423818258)

[Hình 2‑3 Cấu hình chính sách bảo mật 11](#_Toc423818259)

[Hình 2‑4 Các loại file hỗ trợ 12](#_Toc423818260)

[Hình 2‑5 Giới hạn luật mã hóa 12](#_Toc423818261)

[Hình 2‑6 Device Control Setting 14](#_Toc423818262)

[Hình 2‑7 Channel Configuration 15](#_Toc423818263)

[Hình 2‑8 Action 15](#_Toc423818264)

[Hình 2‑9 Agent Management 16](#_Toc423818265)

[Hình 2‑10 Data Identifiers 16](#_Toc423818266)

[Hình 2‑11 Quản lý key 21](#_Toc423818267)

[Hình 2‑12 Lưu lịch sử hoạt động 22](#_Toc423818268)

[Hình 2‑13 Mã hóa file dạng Zip 22](#_Toc423818269)

[Hình 2‑14 Mã hó ổ đĩa 22](#_Toc423818270)

[Hình 2‑15 Xem thông tin file mã hóa 23](#_Toc423818271)

[Hình 2‑16 Chia sẻ file mã hóa 23](#_Toc423818272)

[Hình 3‑1 Hook Server can thiệp vào 2 ứng dụng [14] 30](#_Toc423818273)

[Hình 3‑2 Sử dụng CreateRemoteThread() tải DLL vào ứng dụng. 32](#_Toc423818274)

[Hình 3‑3 User mode và Kernel mode trong Windows [14] 34](#_Toc423818275)

[Hình 3‑4 Quá trình chặn hàm API. 35](#_Toc423818276)

[Hình 3‑5 PE format khi thực thi chương trình [14] 36](#_Toc423818277)

[Hình 3‑6 Mã hóa dữ liệu của Microsoft Word 37](#_Toc423818278)

[Hình 3‑7 Giao diện mới khi lock file 38](#_Toc423818279)

[Hình 3‑8 InstallShield Limited Edition 39](#_Toc423818280)

[Hình 3‑9 InstallShield Limited Edition Download 40](#_Toc423818281)

[Hình 3‑10 InstallShield Limited Edition Error 40](#_Toc423818282)

[Hình 3‑11 Cài đặt WordAddin1 41](#_Toc423818283)

[Hình 3‑12 Cài đặt WordAddin1 41](#_Toc423818284)

[Hình 3‑13 Word Options – Trust Center 42](#_Toc423818285)

[Hình 3‑14 Trust Center – Add-ins 42](#_Toc423818286)

[Hình 3‑15 Quản lý Add-ins trong Word 43](#_Toc423818287)

[Hình 3‑16 Giao diện ban đầu 49](#_Toc423818288)

[Hình 3‑17 Chương trình thông báo khi máy tính nhận USB 49](#_Toc423818289)

[Hình 3‑18 Chương trình thông báo khi USB không tương tác 49](#_Toc423818290)

[Hình 3‑19 Chương trình thông báo khi có USB tương tác 53](#_Toc423818291)

[Hình 3‑20 Chương trình thông báo khi USB không còn tương tác 54](#_Toc423818292)

[Hình 3‑21 Thay thế một USB khác 54](#_Toc423818293)

[Hình 4‑1 Cơ chế mã hóa Stream Cipher. 57](#_Toc423818294)

[Hình 4‑2 Cơ chế mã hóa và giải mã. 58](#_Toc423818295)

[Hình 4‑3 Quá trình tạo khóa mã hóa K. 59](#_Toc423818296)

[Hình 4‑4 Lưu thông tin mã hóa vào đầu file mã hóa. 62](#_Toc423818297)

[Hình 4‑5 Lưu thông tin mã hóa vào cuối file mã hóa. 62](#_Toc423818298)

[Hình 5‑1 Kiến trúc hệ thống. 65](#_Toc423818299)

[Hình 5‑2 Phân hệ can thiệp hệ thống. 66](#_Toc423818300)

[Hình 5‑3 Hoạt động can thiệp hệ thống. 67](#_Toc423818301)

[Hình 5‑4 Quá trình can thiệp hàm API. 68](#_Toc423818302)

[Hình 5‑5 Quá trình cập nhật dữ liệu của file .docx 73](#_Toc423818303)

[Hình 5‑6 Phân hệ bảo vệ dữ liệu. 75](#_Toc423818304)

[Hình 5‑7 Quá trình tạo khóa mã hóa K. 76](#_Toc423818305)

[Hình 5‑8 Phân hệ tương tác với người dùng. 78](#_Toc423818306)

DANH MỤC CÁC BẢNG

**No table of figures entries found.**

TÓM TẮT KHÓA LUẬN

Nội dung khóa luận tìm hiểu thực trạng thất thoát dữ liệu hiện nay. Tỷ lệ dữ liệu thất thoát trên máy tính và các thiết bị lưu trữ là ngày càng lớn. Những thông số trong báo cáo đặt ra nhu cầu cần có những biện pháp bảo vệ dữ liệu hiệu quả trong bối cảnh bùng nổ công nghệ thông tin.

Khảo sát giải pháp bảo vệ dữ liệu của các tổ chức bảo mật, rút ra các đặc điểm cần khắc phục về mặt tương tác với người dùng, giá cả, khó khăn khi cài đặt và triển khai… Dựa trên các mặt còn hạn chế của các phần mềm đi trước, nghiên cứu tạo ra một sản phẩm phần mềm thân thiện và tạo cảm giác thoải mái cho người dùng đã được hiện thực hóa thành phần mềm.

Tìm hiểu các cách can thiệp hệ thống, đánh giá và chọn ra một giải pháp thích hợp áp dụng trong sản phẩm phần mềm. Giải pháp này cần đảm bảo hiệu suất xử lý và giảm thiểu các thao tác dư thừa nhằm tối ưu cho phần mềm.

Đề xuất một giải pháp bảo vệ dữ liệu đơn giản nhưng vẫn phải đảm bảo tính bảo mật. Các thao tác mã hóa và giải mã đơn giản cần tối ưu giảm thời gian xử lý, tạo cho người dùng cảm giác như đang thao tác trên máy tính bình thường.

Cuối cùng, từ những nghiên cứu đạt được, hiện thực hóa giải pháp thành một sản phẩm phần mềm cung cấp đầy đủ chứng năng chính mã hóa và giải mã dữ liệu.

Nội dung khóa luận bao gồm 6 chương:

**Chương** 1**:** Mở đầu

**Chương** 2**:** Tổng quan về các giải pháp ngăn ngừa thất thoát dữ liệu

**Chương** 3**:** Can thiệp hệ thống

**Chương** 4**:** Bảo vệ dữ liệu

**Chương** 5**:** Kiến trúc hệ thống

**Chương** 6**:** Kết luận

# Mở đầu

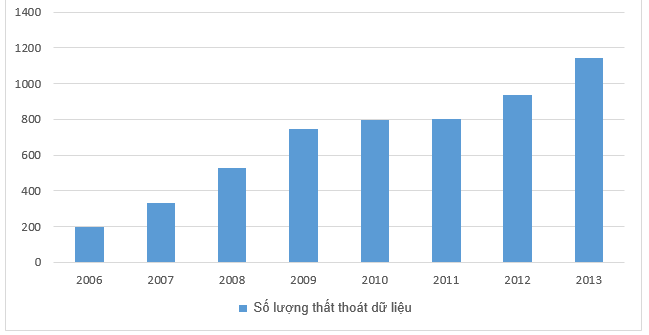
* Nội dung Chương 1 giới thiệu tổng quan về đề tài, phân tích nhu cầu thực tế và lý do thực hiện đề tài. Trên cơ sở đó, nội dung Chương 1 trình bày mục tiêu và nội dung thực hiện của đề tài.

## Giới thiệu

***Ngăn ngừa thất thoát dữ liệu*** (**D**ata **L**oss **P**revention hoặc **D**ata **L**eakage **P**revention – DLP) là một trong những vấn đề quan trọng trong lĩnh vực an toàn thông tin. Trong bối cảnh công nghệ thông tin phát triển mạnh mẽ, dữ liệu được lưu trữ và trao đổi qua nhiều phương tiện khác nhau cũng tăng vọt, kéo theo nhu cầu phát hiện và ngăn ngừa thất thoát dữ liệu ngày càng dược quan tâm.

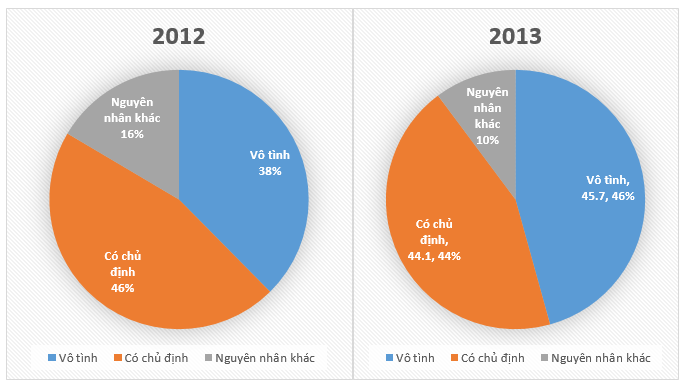
Ổ đĩa USB, máy nghe nhạc MP3, đĩa CD, DVD, và phương tiện di động có khả năng lưu trữ khác rất tiện ích, nhưng cũng đặt ra một mối đe dọa thực sự đối với cá nhân và tổ chức sử dụng chúng. Với kích thước nhỏ và dung lượng lưu trữ rất lớn làm cho các thiết bị này dễ dàng chứa **các dữ liệu mật** của khách hàng và sở hữu trí tuệ để đi thẳng ra cửa trước và rơi vào tay người xấu. Nguyên nhân có thể là do thông qua sự mất mát hoặc trộm cắp. Làm thế nào để bạn biết ai là người lưu trữ những gì trên thiết bị? Và thậm chí nếu người hoặc những người có quyền sử dụng các dữ liệu của bạn, làm thế nào bạn có thể chắc chắn họ đang giữ nó an toàn?

Báo cáo của **InfoWatch** cho thấy mức độ nghiêm trọng của DLP. InfoWatch là tổ chức thu thập và phân tích thất thoát dữ liệu hàng năm. Cụ thể, thất thoát dữ liệu qua từng năm ngày càng tăng cả về số lượng lẫn phương thức. Năm 2012, có tới **934** bản thông tin mật bị thất thoát, tăng 16% so mới năm trước đó [1]. Số lượng này tăng 22% vào năm 2013, tức là có **2143** bản thông tin mật bị mất. Nửa đầu năm 2014, số thông tin rò rỉ đã tăng 32% so với cùng kỳ năm 2013 [1] [2] [3].



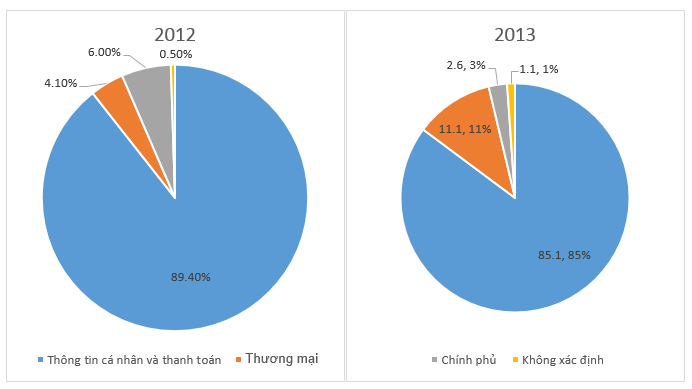
Hình ‑ Số lượng thông tin rò rỉ từ năm 2006-2013 [2]

Thông qua báo cáo của InfoWatch, những dữ liệu thất thoát tập trung vào 3 nguyên nhân chính: Vô tình (Accidental), Có chủ định (Indentional) và các nguyên nhân khác (Unspecified) [1] [2].



Hình ‑ Tỷ lệ thông tin rò rỉ vô tình và có chủ ý năm 2012 và 2013 [2]

Trong khối lượng thông tin rò rỉ khổng lồ đó, thông tin cá nhân và thông tin thanh toán chiếm tỷ lệ nhiều nhất (gần 90%) [1] [2] [3].

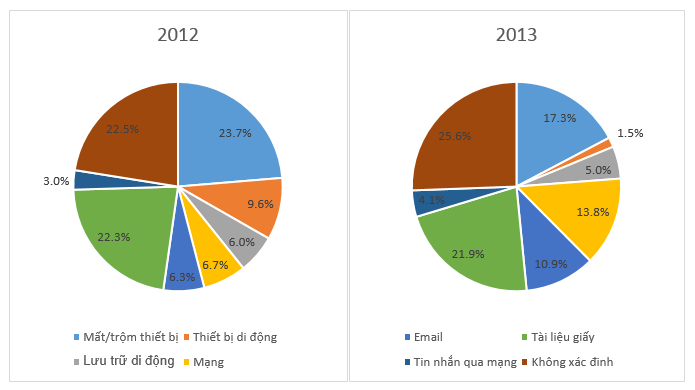


Hình ‑ Tỷ lệ kiểu thông tin bị thất thoát [2]

Những số liệu trên Hình 1‑2 và Hình 1‑3 cho thấy một phần lớn dữ liệu bị thất thoát là do sự vô ý của cá nhân.

Cuối cùng, các phương thức thất thoát dữ liệu cũng rất đa dạng. Bị đánh cắp/mất thiết bị điện tử, thiết bị di động và các thiết bị thông tin di động là những kênh rò rỉ dữ liệu liên quan đến vấn đề cần mã hóa thông tin.

Các thiết bị lưu trữ thông tin hiện nay ngày càng nhỏ gọn về kích thước nhưng dung lượng bộ nhớ lại tăng cao. Chính vì vậy, số lượng lớn dữ liệu quan trọng dễ dàng rõ rỉ ra ngoài qua các thiết bị lưu trữ di động. Thống kê dưới đây cho ta cái nhìn cụ thể hơn về vấn đề này.



Hình ‑ Tỷ lệ rò rỉ dữ liệu theo các kênh [1]

Hình 1‑4 cho thấy tỷ lệ thất thoát dữ liệu qua các thiết bị lưu trữ là không hề nhỏ. Năm 2012, có tới 23,7% là do mất/trộm thiết bị và 13,6% là do các thiết bị lưu trữ di động. Tỷ lệ rò rỉ dữ liệu theo các kênh ở hình Hình 1‑4 trên cho thấy: nếu có thể mã hóa thông tin trên các thiết bị như Laptop, USB… sẽ là một biện pháp hữu hiệu ngăn chặn thất thoát dữ liệu. Những dữ liệu này thường là do vô ý đánh mất dữ liệu như theo báo cáo năm 2013 của InfoWatch:



Hình ‑ Tỷ lệ có chủ ý và vô tình rò rỉ dữ liệu theo kênh năm 2013 [2]

Theo số liệu từ Hình 1‑5, tỷ lệ thất thoát dữ liệu trên thiết bị lưu trữ di động do vô tình cao hơn nhiều so với tỷ lệ có chủ ý. Do vậy, việc nghiên cứu và phát triển giải pháp nhằm ngăn chặn thất thoát dữ liệu do vô tình là rất cần thiết.

## Lý do thực hiện đề tài

Nhận thấy tình trạng thất thoát dữ liệu ngày càng tăng. Chúng em muốn nghiên cứu một giải pháp phần mềm giúp bảo vệ dữ liệu trên máy tính và các thiết bị lưu trữ di động. Dựa trên giải pháp đề ra, hiện thực hóa sản phẩm phần mềm.

Hiện nay, nhiều tổ chức và công ty bảo mật đã cho ra các phần mềm bảo vệ dữ liệu. Nhưng những phần mềm này không mang lại cho người dùng sự thoải mái và tự nhiên. Các thao tác có phần phức tạp và rườm rà, có phần cưỡng chế người sử dụng ở một số phần mềm. Chúng em muốn có một phần mềm “trong suốt” với người sử dụng, tạo cho người dùng cảm giác tự nhiên nhất, giống như đang làm việc trên một máy tính bình thường.

## Mục tiêu đề tài

1. Tìm hiểu về tổng quan Data Leakage Prevention.
   1. Khái niệm Data Leakage Prevention.
   2. Tình hình thất thoát dữ liệu hiện nay.
2. Khảo sát đánh giá các giải pháp ngăn ngừa thất thoát dữ liệu.
   1. Thống kê các giải pháp bảo vệ dữ liệu hiện nay.
   2. Phân tích các đặc điểm của các phần mềm.
3. Tìm hiểu các giải pháp giám sát tập tin ở mức hệ thống.
   1. Các giải pháp chèn DLL vào tiến trình khác.
   2. Các phương pháp chặn hàm API.
   3. Tìm hiểu Microsoft Office Add-in.
   4. Phát hiện thiết bị lưu trữ tương tác với máy tính.
4. Đề xuất giải pháp bảo vệ dữ liệu.
   1. Cơ chế mã hóa và giải mã
   2. Cơ chế phát sinh khóa.
   3. Các vấn đề liên quan đến thông tin mã hóa.
5. Xây dựng mô-đun mã hóa và giải mã.
6. Xây dựng phần mềm ứng dụng hiện thức hóa giải pháp bảo vệ.
   1. Chức năng mã hóa và giải mã.
   2. Chức năng chứng thực người dùng.
   3. Chức năng chia sẻ dữ liệu trên Active Domain.
7. Thử nghiệm trên các môi trường.
8. Xây dựng báo cáo.

## Nội dung luận văn

Luận văn bao gồm 6 chương, nội dung chính từng chương như sau:

**Chương** 1**:** Mở đầu

Trình bày sơ lược Data Leakage Prevention – DLP, tình hình thất thoát dữ liệu hiện nay. Nhu cầu bảo vệ dữ liệu trên các thiết bị lưu trữ di động.

**Chương** 2**:** Tổng quan về các biện pháp ngăn ngừa thất thoát dữ liệu

Khảo sát và thống kê các giải pháp, sản phẩm phần mềm của các tổ chức bảo mật uy tín hiện nay. Phân tích các đặc điểm còn hạn chế của những sản phẩm này và đưa ra các cải tiến cho sản phẩm của nhóm.

**Chương** 3**:** Can thiệp hệ thống

Trình bày các phương pháp can thiệp hệ thống. Dựa trên các nghiên cứu đi trước của các tác giả về kỹ thuật hook API [4] [5], nghiên cứu và áp dụng vào sản phẩm phần mềm.

**Chương** 4**:** Bảo vệ dữ liệu

Nội dung Chương 4 trình bày giải pháp bảo vệ dữ liệu do nhóm tự đề xuất. Giải pháp bảo vệ đơn giản nhưng đảm bảo tính bảo mật áp dụng cho người dùng cá nhân và trên một Active Domain. Ngoài ra, giải pháp có cơ chế chia sẻ dữ liệu giữa các user trong một domain.

**Chương** 5**:** Kiến trúc hệ thống

Nội dung Chương 5 trình bày kiến trúc hệ thống của giải pháp phần mềm. Những thành phần của hệ thống hoạt động và tương tác với nhau được thể hiện chi tiết trong chương này.

**Chương** 6**:** Kết luận và hướng phát triển

Nội dung của Chương 6 trình bày các kết quả đã đạt được và hướng phát triển của đề tài.

# Tổng quan về các biện pháp ngăn ngừa thất thoát dữ liệu

* Nội dung Chương 2 giới thiệu tổng quan về các biện pháp ngăn ngừa thất thoát dữ liệu của các tổ chức bảo mật uy tín. Khảo sát các phần mềm tương ứng được hỗ trợ cho người dùng bảo vệ dữ liệu

## McAfee

### Giải pháp phần mềm

Đối với các công ty kinh doanh nhỏ ( Small business) thì có các loại sản phẩm có các chức năng liên quan đến **DLP**. ( McAfee DLP Endpoint) [5]

1. Endpoint Protection Essential for SMB giá thành từ $22.10 - $24.65 trong 1 năm cho một license được thiết kế cho các công ty kinh doanh nhỏ với 250 thiết bị hoặc ít hơn.
2. Endpoint Protection Advanced for SMB giá thành từ $44.20 - $50.15 cho một license trong 1 năm tương tự như Endpoint Protection Essential nhưng được tích hợp thêm một số tính năng khác như bảo vệ giám sát các thông tin trên các thiết bị mobile như android, mã hóa các thông tin trên các thiết bị …
3. Endpoint Protection Suite giá thành từ $33.69- $48.12 cho một license trong 1 năm được sử dụng tích hợp trong các giải pháp an ninh.
4. Endpoint Protection Advanced. $69.34 - $99.07 cho một license trong 1 năm tương tự như Endpoint Protection Suite và được tích hợp thêm các chức năng bảo vệ khác.

Đối với các doanh nghiệp lớn thì có các bộ sản phẩm khác tương tự cũng tích hợp McAfee Device Control, McAfee Application Control … (**DLP** ) để bảo vệ ngăn chặn các sự cố thất thoát dữ liệu. Giá thành các sản phẩm này sẽ được thỏa thuận giữa reseller và doanh nghiệp [6]. Hai phiên bản cho các doanh nghiệp lớn:

* McAfee Complete Endpoint Protection – Enterprise
* McAfee Complete Endpoint Protection – Business

### Giải pháp phần cứng (thường có giá thành trên $12000)

* + McAfee DLP 4400 Copper Appliance: ứng dụng DLP 4400 là nền tảng Intel-based platform với 6 nhân CPU, 24 GB RAM và hơn 8 TB lưu trữ trong thành phần 2U.
  + McAfee DLP 5500 Copper Appliance: ứng dụng DLP 4400 là nền tảng Intel-based platform với 6 nhân CPU, 32 GB RAM và hơn 10 TB lưu trữ trong thành phần 2U.
  + Ứng dụng giải pháp phần cứng DLP là tích hợp trong thiết bị. Không cần yêu cầu phần mềm, phần cứng hoặc cơ sở dữ liệu kèm theo.

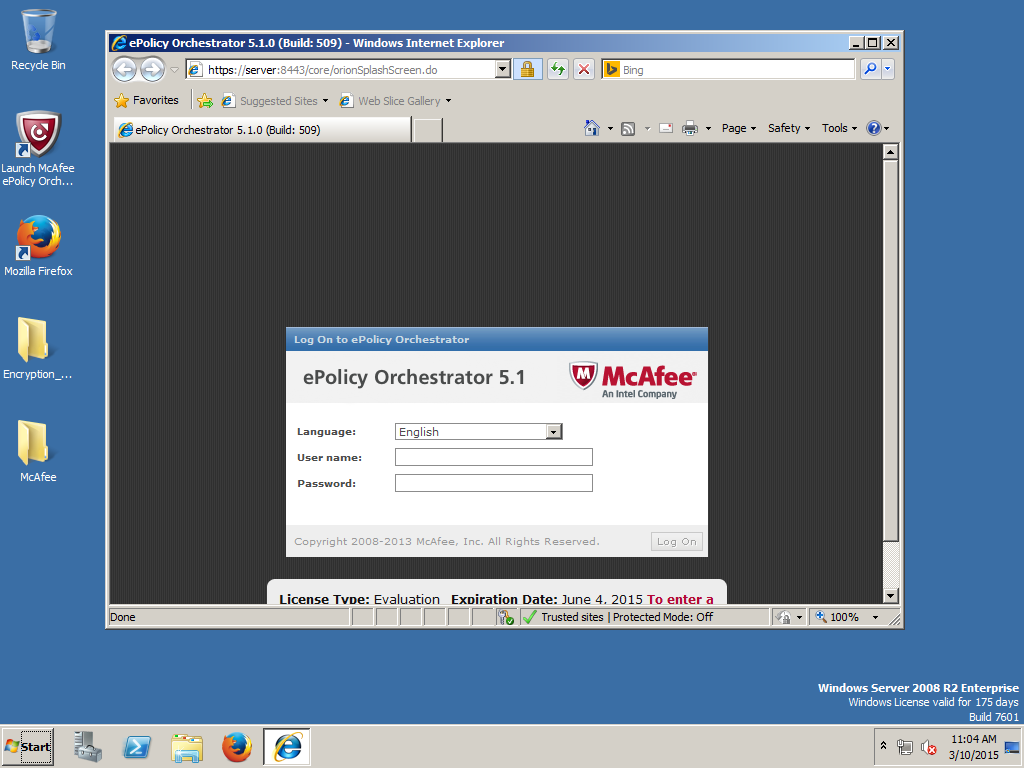
### Các tính năng chung của McAfee DLP Endpoint

**Protect and Comply**: Theo dõi các sự kiện thời gian thực và áp dụng các chính sách an ninh một cách nhanh chóng và dễ dàng để điều chỉnh và hạn chế việc nhân viên sử dụng và truyền tải dữ liệu nhạy cảm mà không ảnh hưởng đến năng suất lao động. Bảo vệ dữ liệu khỏi các mối đe dọa xuất phát từ bên trong như email, IM, các trang web, sao chép dữ liệu qua USB và công việc in ấn. Ngoài ra còn có thể ngăn chặn mất mát các dữ liệu mất khỏi các chương trình Trojans, worms và các ứng dụng chia sẻ tập tin khác.

**Control and Manage**: Công cụ quản lý thiết bị toàn diện giúp cho ta kiểm soát và ngăn chặn các dữ liệu mật được sao chép vào USBs, ổ đĩa flash, CD/ DVD, iPod và các thiết bị lưu trữ di động khác. Các thông số thiết bị như product ID, vendor ID, số serial, loại thiết bị, và tên thiết bị có thể được xác định và phân loại. Hơn nữa, các chính sách khác nhau như chặn hoặc mã hóa có thể được áp dụng cho nội dung các thông tin được tải lên các thiết bị.

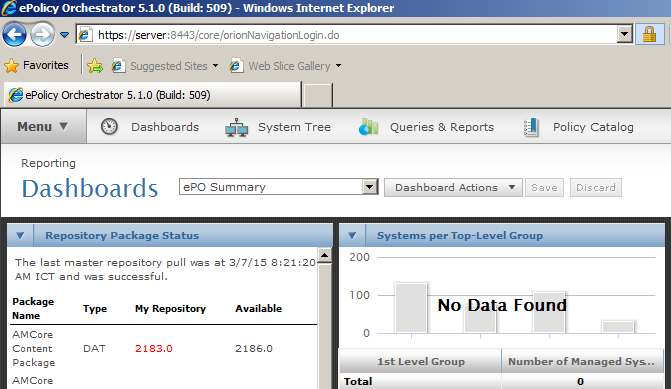
**Centralized Management Through McAfee ePO Software**: Tích hợp với phần mềm McAfee EPO cung cấp giám sát sự kiện thời gian thực và chính sách tập trung và quản lý sự cố. Nó cho phép dễ dàng thu thập các dữ liệu sử dụng quan trọng như người gửi, người nhận, thời gian và các chứng cứ dữ liệu. Với một cú click chuột, phần mềm McAfee cho ra các báo cáo chi tiết để chứng minh với các kiểm toán viên, quản lý cấp cao, và các bên liên quan cấp rằng các biện pháp và quy định bên trong đã được áp dụng đúng chỗ.

### Trial McAfee Complete Endpoint Advanced



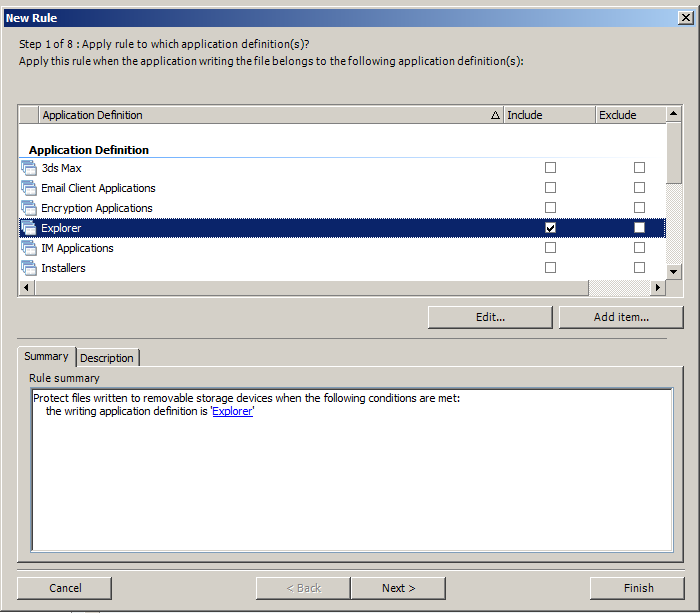
Hình ‑ Giao diện đăng nhập

Hình 2‑1 hiển thị giao diện đăng nhập trên trình duyệt. Phần mềm yêu cầu User name và Password kèm theo tùy chọn ngôn ngữ.



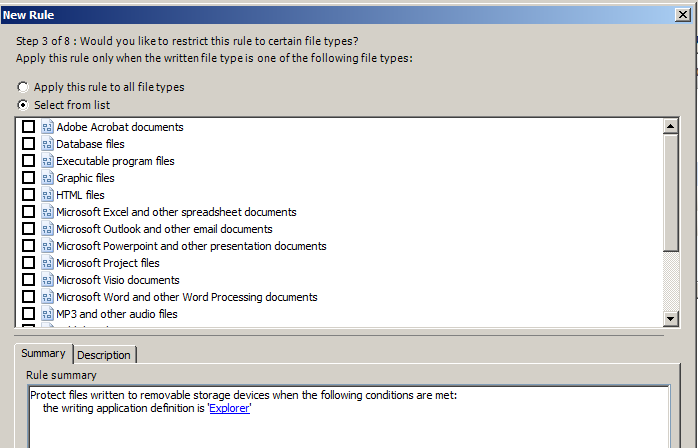
Hình ‑ Dashboard

Sau khi đăng nhập, ứng dụng chuyển đến màn hình Dashboard như Hình 2‑2



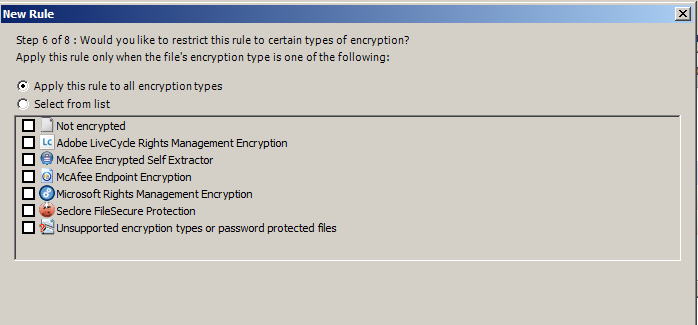
Hình ‑ Cấu hình chính sách bảo mật

Người dùng cấu hình chính sách bảo mật trên cửa sổ trên Hình 2‑3.



Hình ‑ Các loại file hỗ trợ.

Trong Hình 2‑4, cửa sổ giao diện cho thấy ứng dụng cho phép tùy chọn các loại file hỗ trợ bảo vệ.



Hình ‑ Quy định kiểu mã hóa.

Hình 2‑5 cho thấy một điểm hay của ứng dụng là cho phép chọn kiểu mã hóa.

## TrendMicro

### Giải pháp phần mềm

Các giải pháp phần mềm hiện tại hầu hết của Trend Micro đa số đều được tích hợp thêm DLP plug-in.

Đối với các SMB thì có các gói sản phẩm Worry-Free [7]:

* TrendMicro Worry-Free Business Security Services $29.06/user (100 user)
* Trend Micro Worry-Free™ Business Security Standard $32.36/user (100 user)
* TrendMicro Worry-Free Business Security Advanced $59.32/ user (100 user)

Nếu như các gói này được mua với số lượng từ 2-100 thì sẽ có các giá thành khác nhau. Ví dụ như gói TrendMicro Worry-Free Business Security Advanced sẽ có giá $124.04/year với 2 người dùng còn với 100 users giá sẽ là $5932/year.

Đối với các doanh nghiệp và midsize thì có nhiều các giải pháp khác nhau với các giá thành thương lượng [8] giữa nhà cung cấp và doanh nghiệp như:

* Cloud and Data Center Security
* Complete User Security
* Custom defend

### Các tính năng của các sản phẩm tích hợp Trend Micro Intergrated Data Loss Prevention

Với một plug-in “nhẹ”, bạn có thể đạt được tầm nhìn, kiểm soát các dữ liệu nhạy cảm và có thể ngăn chặn các thất thoát dữ liệu qua USB, email và web một cách dễ dàng, nhanh chóng. Plug-in DLP không yêu cầu các phần cứng hoặc hay phần mềm nào khác, và đó là đòn bẩy để xây dựng lĩnh vực và it leverages built-in regional and industry-specific templates to simplify deployment. Phần mềm tích hợp DLP cho phép chúng ta triển khai an toàn thông tin chỉ với một phần nhỏ chi phí và thời gian so với các giải pháp DLP truyền thống của các doanh nghiệp khác [9].

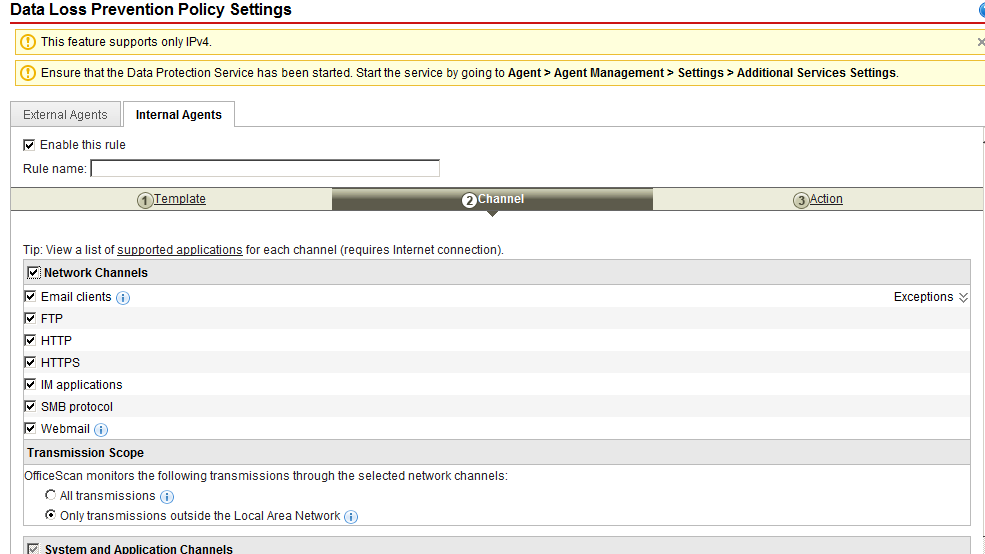
* Theo dõi mạng 24/7 với việc điều chỉnh theo thời gian thực
* Theo dõi và ghi lại các dữ liệu nhạy cảm truyền qua các điểm mạng
* Cho phép các IT hạn chế việc sử dụng các ổ đĩa USB, các thiết bị di dộng cổng USB, ghi đĩa CD/DVD, và các thiết bị di động khác.
* Theo dõi và xử lý với các dữ liệu sử dụng không đúng dựa trên từ khóa, regular expression và thuộc tính tập tin.
* Cải thiện cái nhìn và quyền kiểm sát với giải pháp tích hợp đầy đủ, quyền điều hành tập trung.
* Tốc độ kiểm tra và thực thi nắm bất dữ liệu và báo cáo thời gian thực.

### Trial TrendMicro



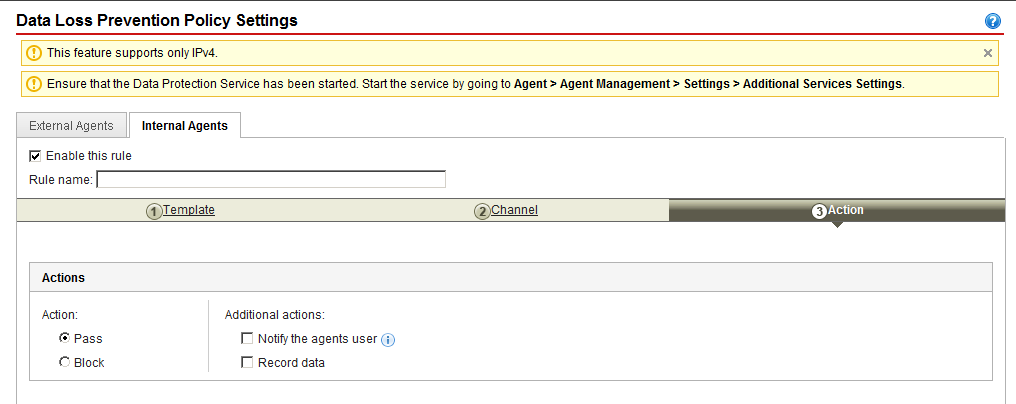
Hình ‑ Device Control Setting

Trong Hình 2‑6, ứng dụng cho phép tùy chọn quyền truy cập dữ liệu.



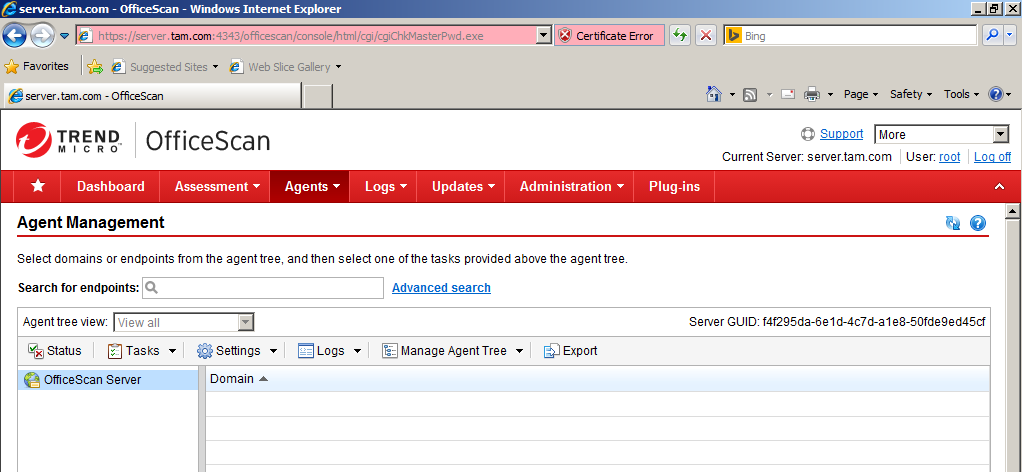
Hình ‑ Cấu hình trên các kênh mạng

Như Hình 2‑7 cho thấy, ứng dụng còn có thể áp dụng trên các phương giao thức mạng.



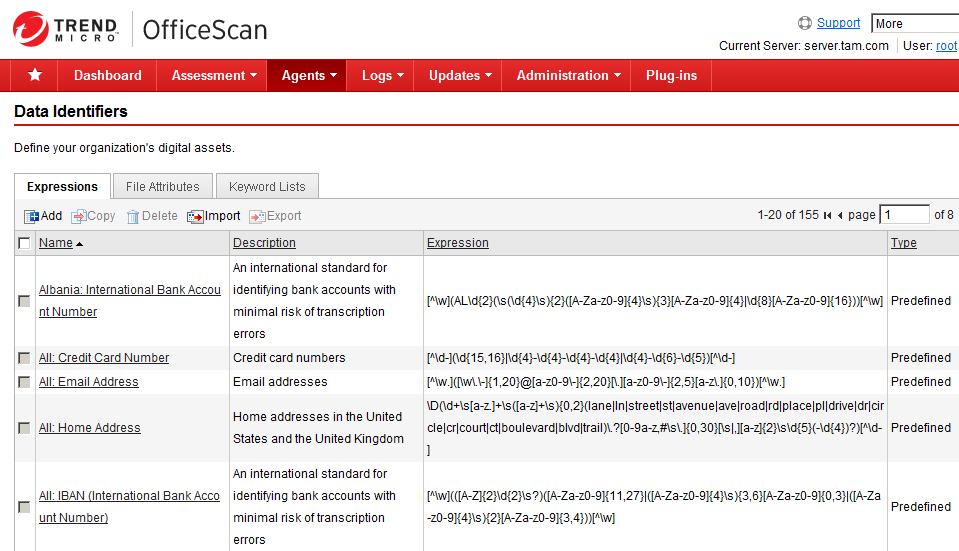
Hình ‑ Action

Phần cuối trong cấu hình là lựa chọn cách ứng xử với các cấu hình trên. Hình 2‑8 hiển thị các lựa chọn cho phép hoặc ngăn chặn. Ngoài ra, còn có thêm các lựa chọn như thông báo cho người dùng và lưu bản ghi dữ liệu.



Hình ‑ Giao diện quản lý chính.

Giao diện quản lý chính của ứng dụng mô tả trong Hình 2‑9. Ứng dụng cho phép quản lý trên domain hoặc endpoints với các tùy chọn kiểm soát dữ liệu.



Hình ‑ Data Identifiers

Quy định các tài sản của công ty với các thuộc tính như tên gọi, mô tả,.. để dễ dàng quản lý. Hình 2‑10 hiển thị một ví dụ về Data Identifiers.

## Symantec

### Giải pháp phần mềm

Symantec cung cấp rất nhiều phần mềm cho giải pháp bảo mật và lưu trữ. Những phần mềm này dễ dàng tích hợp với nhau [10].

* **Symantec Backup Exec System Recovery** – Data Loss Prevention tích hợp với Backup Exec System Recovery để quét các bản dự phòng (backup) dữ liệu quan trọng.
* **Symantec Control Compliance Suite** – Data Loss Prevention dữ liệu sự cố có thể được nạp vào Control Compliance Suite để xác định hệ thống có phải chịu trách nhiệm về chính sách kiểm soát kỹ thuật dựa trên các dữ liệu được lưu trữ trong đó.
* **Symantec Encryption** – Data Loss Prevention tích hợp với **Symantec Endpoint Encryption** để thực thi mã hóa dựa trên chính sách của các tập tin khi chúng được sao chép trong thời gian thực tới USBs; nó tích hợp với **Symantec FileShare Encryption** để giải nén, giải mã và phân tích văn bản trong các tài liệu mã hóa chia sẻ; nó tích hợp với **Symantec Universal Gateway Email** để thực thi mã hóa dựa trên chính sách của email và cung cấp xác nhận vòng kín bảo mật tại Data Loss Prevention Enforce Platform.
* **Symantec Endpoint Protection** – Data Loss Prevention tích hợp với Endpoint Protection để thực thi dựa trên chính sách lockdown (e.g., application control, port blocking, device control) trên máy tính và laptop. Với 1 năm hỗ trợ dành cho doanh nghiệp nhỏ, tương ứng 1 endpoint là 30 USD. Với 2 năm hỗ trợ, 1 endpoint tương ứng 50 USB. Con số này là 60 USD với 3 năm hỗ trợ.
* **Symantec Enterprise Vault** – Powered by the Data Loss Prevention công cụ phát hiện nội dung, **Enterprise Vault Data Classification Services** cung cấp dịch vụ dữ liệu phân loại phân loại dữ liệu và dịch vụ gắn thẻ chính sách định hướng cho email lưu trữ. Symantec Enterprise Vault Compliance Accelerator và Symantec Enterprise Vault Discovery Accelerator tận dụng các kết quả phân loại để hỗ trợ lưu trữ hồ sơ, tuân thủ pháp luật, và eDiscovery.
* **Symantec Messaging Gateway** – Data Loss Prevention tích hợp với Messaging Gateway vì vậy bạn có thể dễ dàng xem lại, phát hành và tuyến đường đi tin nhắn trực tiếp từ các Data Loss Prevention Enforce Platform.
* **Symantec Mobile Management** – Data Loss Prevention tích hợp với Mobile Management để thực thi các thiết lập mạng riêng ảo trên các thiết bị di động, màn hình hiển thị thông báo trên màn hình khi người dùng cố gắng để làm xáo trộn với hồ sơ thiết bị, và thực thi các điều khiển từ xa dựa trên chính sách trên các thiết bị.

### Các tính năng chống mất mát dữ liệu [11]

**Managerment and Reporting**

90% của DLP là về những thao tác sau khi bạn tìm dữ liệu quan trọng. Với Symantec Data Loss Prevention Enforce Platform, bạn có thể dễ dàng quản lý các chính sách và quy trình khắc phục hậu quả, xem xét sự cố, và giảm thiểu thiệt hại từ một nền tảng quản lý web thống nhất.

Enforce Platform là một giao diện điều khiển quản lý dựa trên web mạnh mẽ nơi bạn quản lý các chính sách và quy trình công việc mất mát dữ liệu, xem xét và khắc phục sự cố, phân tích và giảm thiểu rủi ro báo cáo và quản trị hệ thống. Nó bao gồm một module báo cáo nâng cao, Mất Symantec dữ liệu phòng chống IT Analytics, bạn có thể sử dụng dễ dàng tạo ra các báo cáo và biểu đồ mà giao tiếp ROI chương trình DLP của bạn tới giám đốc điều hành, các bên liên quan kinh doanh và kiểm toán viên.

**Laptops and Desktops**

Nhân viên được tải về và gửi dữ liệu bí mật trong khi ở văn phòng, trên đường đi hay ở nhà. Symantec Data Loss Prevention cho Endpoint monitors và bảo vệ dữ liệu được sử dụng trên máy tính xách tay và máy tính để bàn, khi người dùng đang ở trong và ngoài mạng công ty.

Quét Endpoint Khám phá và hàng tồn kho ổ đĩa cứng nội bộ trên máy tính xách tay và máy tính để bàn cho dữ liệu bí mật.

Endpoint Ngăn chặn theo dõi hoạt động của người dùng trên và ngoài mạng công ty; ngăn cản dữ liệu mật khỏi bị sao chép hoặc chia sẻ qua email không phù hợp, lưu trữ di động (ví dụ, USB, CD / DVD), in và fax, và lưu trữ đám mây (ví dụ, Dropbox). Các Endpoint Agent giám sát một loạt các sự kiện người sử dụng trên thiết bị đầu cuối vật lý đang chạy Windows XP, Windows Vista, Windows 7, và bây giờ Windows 8.1; phát hiện ra các dữ liệu được lưu trữ trên Mac OS X; giám sát máy tính để bàn ảo và các ứng dụng được lưu trữ bởi Citrix XenApp 6,5, VMware View và Microsoft Hyper-V; và màn hình dữ liệu chuyển qua Microsoft Remote Desktop Protocol (RDP).

**Mobile Devices**

Nhân viên được mang thiết bị riêng của họ để làm việc và truy cập dữ liệu bí mật từ họ, có hoặc không có sự cho phép của IT Security. Trong thực tế, một nghiên cứu mới tiết lộ rằng 2 trong số 5 nhân viên tải về tập tin làm việc với điện thoại thông minh và máy tính bảng cá nhân của họ. Hiện tại đội an ninh có thể quản lý Mang thiết bị riêng của bạn (BYOD) chính sách trong khi đảm bảo dữ liệu bí mật với Symantec dữ liệu phòng chống mất mát cho Mobile.

Mobile Email Monitor phát hiện email bí mật tải bởi người dùng iPad, iPhone và các thiết bị Android hiện nay thông qua giao thức Microsoft Exchange ActiveSync.

Màn hình điện thoại di động Ngăn chặn và bảo vệ mạng lưới truyền thông ra bên ngoài được gửi từ các khách hàng mail gốc, trình duyệt và các ứng dụng khác (ví dụ, Dropbox, Facebook) trên iPad và iPhone.

**Email và Web**

Email và web là hai trong số các kênh truyền hình phổ biến nhất cho sự mất mát dữ liệu. Một cuộc khảo sát gần đây cho thấy 50 phần trăm nhân viên thường xuyên gửi email tập tin làm việc từ văn phòng đến các tài khoản cá nhân. Symantec dữ liệu phòng chống mất mát cho mạng giám sát điểm ra và email đám mây để ngăn chặn dữ liệu bí mật bị phơi nhiễm qua giao thức mạng.

Ngăn chặn đám mây cho Microsoft Office 365 phát hiện dữ liệu bí mật trong email được gửi từ điện toán đám mây Microsoft Exchange Online. Nó liền mạch kéo dài chính sách mất dữ liệu hiện tại của bạn và công việc, do đó bạn có thể tự tin di chuyển các máy chủ Exchange của bạn để đám mây của Microsoft. Ngăn chặn đám mây cung cấp triển khai linh hoạt tại chỗ hoặc trong một cơ sở hạ tầng như một dịch vụ (IaaS) môi trường. Nó cũng hoàn toàn phù hợp với Symantec Email Security.cloud để đảm bảo cung cấp email thức.

Network Monitor dò tìm dữ liệu bí mật được gửi qua giao thức mạng có nguy cơ cao mà không cần lấy mẫu hoặc thả các gói tin: SMTP, HTTP, FTP, IM, NNTP, giao thức cổng cụ thể tùy chỉnh, và bây giờ Internet Protocol Version 6 (IPv6) mạng. Ngăn chặn cho mạng Email dò tìm dữ liệu bí mật trong email; thông báo cho người sử dụng vi phạm chính sách; và các khối hoặc các tuyến đường để gửi email mã hóa các cổng giao an toàn. Nó hỗ trợ tích hợp với bất kỳ chất độc SMTP-compliant Mail Transfer (MTA) và các dịch vụ đám mây như Microsoft Exchange Online và Symantec Email Security Cloud.

Ngăn chặn cho mạng Web dò tìm dữ liệu bí mật được gửi qua HTTP và HTTPS; thông báo cho người sử dụng vi phạm chính sách; và khối hoặc có điều kiện loại bỏ dữ liệu từ bài viết web. Nó hỗ trợ tích hợp với bất kỳ proxy Web ICAP tuân thủ và các dịch vụ đám mây như Google Postini và Symantec Web Security.coud.

**File Shares, Databases and Document Repositories**

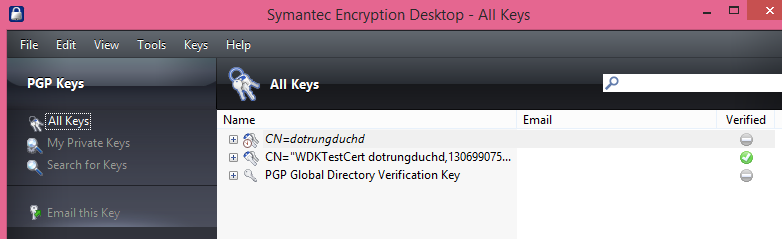
Người dùng được lưu trữ một lượng lớn các tập tin bí mật trên mạng công ty với quyền truy cập mở, mà làm cho họ dễ bị thiệt hại và trộm cắp. Symantec dữ liệu phòng chống mất cho lưu trữ quét trung tâm dữ liệu của bạn để khám phá và bảo vệ dữ liệu bí mật được lưu trữ trên tập tin chia sẻ, cơ sở dữ liệu, và các kho chứa. Dữ liệu Insight là một công nghệ giám sát tập tin duy nhất mà phân tích các mẫu truy cập và sử dụng tập tin trên mạng lưu trữ đính kèm (NAS) filers, các máy chủ Windows, và SharePoint. Nó xác định chủ sở hữu dữ liệu true; tính toán rủi ro folder để khắc phục được ưu tiên; tương quan chủ dữ liệu với sự cố lưu trữ; và cảnh báo bạn để hoạt động và người sử dụng outlier bất thường.

Khám phá mạng lưới thực hiện tốc độ cao chức năng quét của máy chủ tập tin, cổ phiếu, cơ sở dữ liệu, và kho tài liệu bao gồm cả Microsoft SharePoint và SharePoint Online, Documentum và LivelinkBREAK.

Mạng lưới Bảo vệ chặt tập tin bí mật bằng cách ly, di chuyển, hoặc áp dụng mã hóa và quản lý quyền kỹ thuật số dựa trên chính sách để các tập tin và thư mục tiếp xúc. Self-Service Portal Xử lý ô nhiễm cho phép chủ sở hữu dữ liệu kinh doanh để xem xét và khắc phục các vi phạm chính sách tập tin mạng trực tiếp từ một cổng thông tin trực tuyến trực quan, và sắp xếp hợp lý các quy trình xử lý rủi ro.

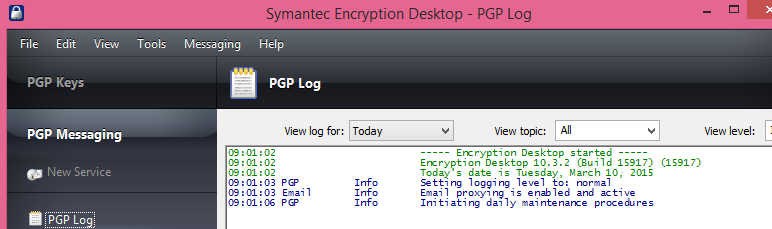
### Trial Symantec File Share Encryption

Giao diện Symantec Encryption Desktop



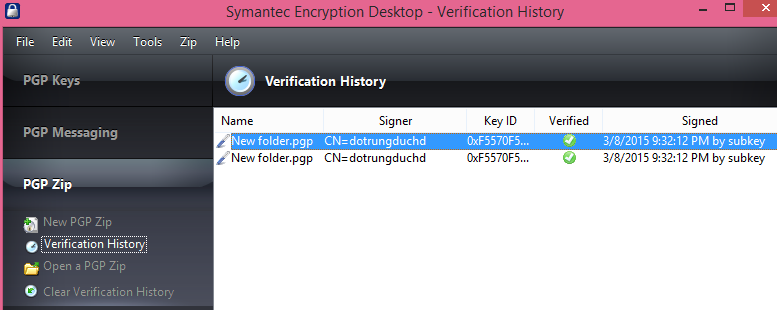
Hình ‑ Giao diện quản lý khóa

Hình 2‑11 hiển thị giao diện quản lý khóa của ứng dụng. Có hỗ trợ khóa bất đối xứng.



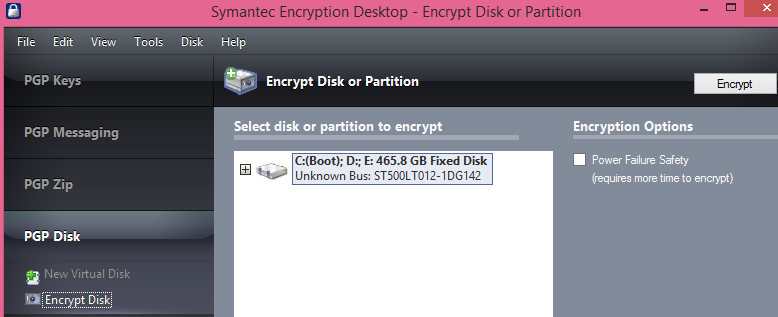
Hình ‑ Lưu lịch sử hoạt động

Giao diện hiển thị lịch sử hoạt động cửa ứng dụng như Hình 2‑12. Tất cả thao tác mã hóa và giải mã được lưu lại, được quả lý theo trình tự thời gian.



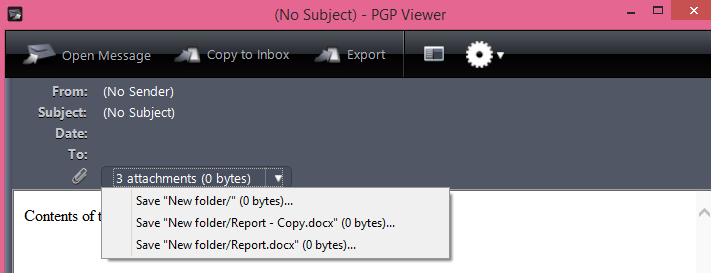
Hình ‑ Mã hóa file dạng Zip

Ví dụ cụ thể, thư mục new folder được mã hóa bởi user dotrungduchd (Hình 2‑13) cho kết quả là 1 file zip mã hóa.



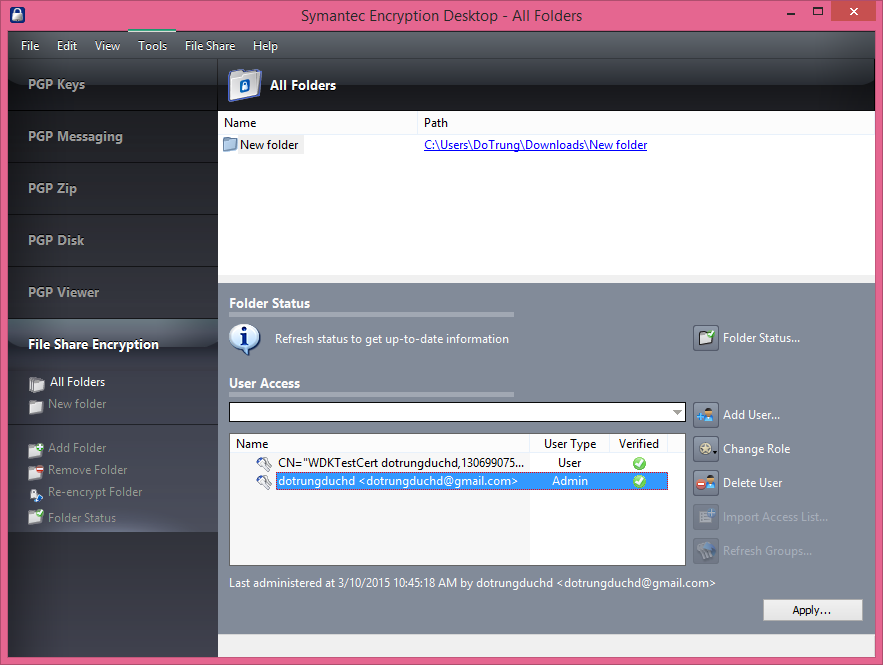
Hình ‑ Chức năng mã hóa ổ đĩa.

Ngoài khả năng mã hóa file và folder, Hình 2‑14 cho thấy ứng dụng có khả năng mã hóa ổ đĩa nhưng thời gian mã hóa có thể khá lâu.



Hình ‑ Xem thông tin file mã hóa

Trong khi ứng dụng đang chạy, nó có thể xem danh sách tập tin trong một file mã hóa như trong Hình 2‑15.



Hình ‑ Chia sẻ dữ liêu mã hóa

Tính năng chia sẻ dữ liệu là tính năng cuối cùng của ứng dụng. Ví dụ trong Hình 2‑16, thư muc new folder đã được mã hóa và chia sẻ cho 2 user, trong đó user dotrungduchd là admin.

## RSA

### Giải pháp phần mềm

Khám phá và theo dõi vị trí và luồng của dữ liệu nhạy cảm như dữ liệu thẻ tín dụng của khách hàng, nhân viên PII, hoặc tài sản trí tuệ của doanh nghiệp. Hướng dẫn người dùng cuối và thực thi kiểm soát để ngăn ngừa mất dữ liệu nhạy cảm thông qua email, web, máy tính, điện thoại thông minh, và nhiều hơn nữa [12].

### Các tính năng chống mất mát dữ liệu [12]

**Tổng quan**

Ngăn chặn việc mất dữ liệu nhạy cảm thông qua nhiều nguy cơ rủi ro, bao gồm email, webmail, phương tiện truyền thông xã hội, FTP, Web, Web 2.0, máy tính, máy ảo, điện thoại thông minh, Microsoft SharePoint, máy chủ tập tin, NAS / SAN, cơ sở dữ liệu, và các thiết bị USB.

**Phân loại chính xác**

Xác định các dữ liệu nhạy cảm với độ chính xác công nghiệp tốt nhất thông qua sự kết hợp RSA DLP của nhận thức khoa học dựa trên phân loại nội dung, vân tay dựa trên máy móc, phân tích siêu dữ liệu phong phú, và các chính sách chuyên gia mục đích xây dựng.

**Giáo dục người dùng**

Dựa vào RSA DLP để theo dõi các hành động người dùng lấy dữ liệu nhạy cảm và giáo dục họ trong thời gian thực về các vi phạm chính sách. Điều này cải thiện nhận thức về nguy cơ trong những người dùng cuối, ảnh hưởng đến hành vi của họ trong việc đối phó với các dữ liệu nhạy cảm.

**Nền tảng tích hợp**

Tích hợp RSA DLP với nhiều nền tảng doanh nghiệp, bao gồm cả Microsoft, Cisco, EMC, VMware, Citrix, McAfee, Symantec, Trend Micro, và Blue Coat, để tối đa hóa việc sử dụng các cơ sở hạ tầng hiện tại của bạn.

**Tự động hóa công việc**

Cải thiện quản lý dự án DLP bằng cách tận dụng một quy trình làm việc tự động người- và quá trình làm trung tâm cho chính sách quản lý, khắc phục sự cố, quản lý và báo cáo.

## WebSense

### Giải pháp phần mềm [13]

**TRITON AP-**WEB: Có được sự bảo về thời gian thực chống lại các Advanced Threats và việc đánh cắp dữ liệu với nhiều phương thức triển khai cho cả on-premise và người dùng từ xa.

* Chặn các Advanced Threats – TRITON AP-WEB nhận dạng các Zero-day malware và các hoạt động gây hại khác trên toàn bộ Kill chain để ngăn chặn các mối đe dọa mới nhất.
* Bảo vệ toàn diện – Thống nhất hệ thống bảo vệ inbound va outbound Web, chặn các mối đe dọa và cung cấp actionable insight.
* Ngăn chặn đánh cắp dữ liệu – Một engine được tích hợp đầy đủ theo chuẩn DLP doanh nghiệp đơn giản hóa việc an ninh thông tin và compliance bao gồm OCR, “Drip DLP” và các tính năng độc đáo khác.
* Cho phép truy cập liên tục tới tài nguyên Web, điều khiển và làm việc với thiết bị di động, biết rằng đó là dữ liệu nhạy cảm còn lại.

TRITON AP-EMAIL: Xác định các mục tiêu bị tấn công, những người dùng có nguy cơ cao và các mối đe dọa bên trong đồng thời cho phép các thiết bị di động làm việc và an toàn như công nghẹ của Microsoft Office 365™.

* Ngăn chặn APT và các mối đe dọa các khác.
* Bảo mật các dữ liệu nhạy cảm khỏi các cuộc tấn công bên ngoài và mối đe dọa bên trong.
* Áp dụng các công nghệ mới như Microsoft Office 365 và hỗ trợ cho các nhân viên từ xa một cách an toàn.
* Giúp do người dùng nâng cao ý thức và xác định các hành động có nguy cơ gây hại.

TRITON AP-DATA: Phát hiện và bảo vệ những dữ liệu nhảy cảm bất cứ nơi nào chứa nó – trên servers, endpoints hoặc các dich vụ đám mây như MO365, cũng như trong sử dụng hay chuyển đổi qua Web hay email gateways.

* Áp dụng dịch vụ đám mây như MO365 và Box mà không sợ bị đánh cắp dữ liệu
* Deploy easy-to-use security controls for compliance and regulatory requirements for auditors as well as for executives and investors.
* Phát hiện các dữ liệu nhạy cảm bên trong hình cũng như trong các dữ liệu được scan và ảnh chụp màn hình dễ dàng.
* Thống nhất các giải pháp bảo mật, các chính sách bảo vệ, chia sẻ thông tin theo nhiều điểm và tập trung quyền kiểm soát cho an toàn thông tin.

TRITON AP-ENDPOINT: Protect roaming người dùng khỏi việc đánh cắp dữ liệu và nắm quyền kiểm soát thông tin nhạy cảm trên các hệ thống Mac OS X và Windows trong và ngoài mạng.

* Phát hiện và bảo vệ các dữ liệu quan trọng khỏi đánh cắp khi người dùng ở trong và ngoài mạng công ty.
* Bảo mật các endpoint Mac OS X và Windows, bao gồm chống thoát dữ liệu qua USB và các thiết bị lưu trữ khác.
* Safely embrace cloud services like social media, Microsoft Office 365™ and Box through data visibility and control.
* Nhanh chóng đáp ứng các yêu cầu cần tuân thủ, quy định với một thư viện rộng lớn đầy các chính sách và thỏa mãn các auditors với các mẫu báo cáo chuẩn hay tùy chỉnh.
* Kiến trúc TRITON cung cấp cách quản lý thống nhất của giải pháp TRITON, củng cố các tasks để các nhân viên IT có thể tập trung cho các dự án khác.

Giá thành của các sản phẩm sẽ được thỏa thuận giữa **WebSense Partner** và người mua.

## Kết luận

Các giải pháp bảo vệ dự liệu trên thiết bị lưu trữ của các công ty bảo mật thường áp dụng theo mô hình server – endpoint (client). Những chính sách bảo mật được server đưa ra và bắt buộc endpoint phải tuân theo. Nhưng chính điều đó gây ra sự mất tự nhiên, cảm giác làm chủ thiết bị của mình không còn. Thay vào đó là một loạt các chính sách cưỡng bức người dùng như chặn USB, chỉ cho phép đọc USB,…

Hơn thế nữa, những giải pháp này áp dụng cho những công ty quy mô lớn, giá thành rất cao. Trong khi cá nhân và các mô hình vừa và nhỏ chưa có giải pháp thay thế nào phù hợp ngoài việc thao tác thủ công mã hóa dữ liệu trước khi lưu trữ ra các thiết bị nhớ ngoài.

Phân tích các đặc điểm của những phần mềm bảo mật giúp nhóm có các ý tưởng hoàn thiện giải pháp bảo vệ dữ liệu.

# Giải pháp giám sát tập tin ở mức hệ thống

* + Nội dung Chương 3 trình bày các biện pháp can thiệp vào hệ thống để chặn và lấy thông tin cần thiết phục vụ cho giải pháp bảo mật sau này. Chương này trình bày 2 nội dụng chính:

- Hook API

- Microsoft Office Add-in

## Hook API

* Vấn đề:

Trong quá trình xử lý, hệ thống gọi các hàm API để thực hiện các chức năng nhất định. Các hàm API khi thực hiện sẽ xuất hiện sự kiện và thông báo tương ứng. Vấn đề chính của chúng ta là làm cách nào để biết được sự kiện khi nào một hàm API được gọi và thực thi? Làm sao để thay đổi hoặc thêm một vài xử lý trước khi hàm kết thúc?

* Giải pháp

Sử dụng Hook API để giải quyết vấn đề. Hook API là kỹ thuật được sử dụng để theo dõi ứng dụng và các hàm API của hệ thống.

Hook API giúp lập trình viên theo dõi và kiểm soát các API được gọi bởi tiến trình đang được can thiệp. Sử dụng để xác định và sửa lỗi, thay đổi và phát triển chức năng của hàm API.

Trong giải pháp này có 2 phần quan trọng đó là chèn DLL vào process (inject) và chặn hàm API (intercept).

## Kỹ thuật chèn DLL vào process

### Registry

* Vấn đề

Làm cách nào để chèn DLL vào một tiến trình đang chạy. Có thể chèn DLL vào các chương trình trong quá trình chúng tải các thư viện liên kết động (DLL)?

* Giải pháp

Trong môi trường Windows, trên thực tế có thể chèn một DLL vào các tiến trình có liên kết với USER32.DLL bằng cách thêm tên của DLL đó vào trong registry: ***HKEY\_LOCAL\_MACHINE\Software\Microsoft\***

***Windows NT\CurrentVersion\Windows\AppInit\_DLLs.***Khi đó những ứng dụng cơ bản của Windows sẽ tải DLL được quy định trong registry lên giống như một phần của quá trình khởi tạo USER32.DLL. Trong hàm DllMain của USER32 thực hiện lệnh gọi hàm LoadLibrary() và tải những DLL có tên trong thanh ghi. Hạn chế của kỹ thuật này là chỉ hỗ trợ cho Windows NT và Windows 2k.

Một số lưu ý:

* Khởi động lại Windows để kích hoạt hoặc dừng chèn DLL vào tiến trình.
* Chỉ có tác dụng với những tiến trình sử dụng USER32.DLL, vì thế có thể một số ứng dụng không thực hiện được.
* Không kiểm soát được quá trình chèn DLL vào tiến trình. Nó có tác dụng với tất cả ứng dụng giao diện GUI (Graphic User Interface) nên có thế có nhiều chi phí phát sinh.

### System-wide Windows Hooks

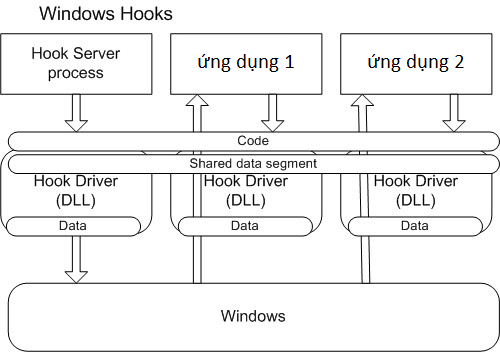
* Vấn đề

Khi chúng ta có nhu cầu xử lý Hook trên toàn bộ hệ thống. Vấn đề đặt ra là làm sao có thể chèn DLL vào tất cả process đang chạy (ngoại trừ các service của windows).

* Giải pháp

Sử dụng System-wide Windows Hooks tác động và inject DLL toàn cục vào tất cả các ứng dụng.

Đây là một kỹ thuật rất phổ biến để chèn DLL vào process. Ứng dụng có thể cài một hàm lọc để theo dõi message trong hệ thống và xử lý chúng trước khi tới windows procedure. Sử dụng hàm **SetWindowsHookEx**() để cài đặt hook cùng các tham số tương ứng. Khi cài đặt hook thành công, hệ điều hành sẽ ánh xạ DLL vào địa chỉ của mỗi process. Những biến trong DLL sẽ được tự xử lý và không thể chia sẻ cho những process khác. Mọi biến chứa dữ liệu cần chia sẻ phải ở trong ‘*share data section*’.



Hình ‑ Hook Server can thiệp vào 2 ứng dụng [14]

Sau khi hàm SetWindowsHookEx() thực hiện thành công, hệ điều hành tự động chèn DLL vào tất cả các process đáp ứng yêu cầu của hook filter. Một khi đã chèn DLL vào trong process, sử dụng hàm UnhookWindowsHookEx() để unload DLL hoặc tắt ứng dụng hook. Khi gọi hàm UnhookWindowsHookEx(), hệ điều hành lặp qua danh sách những process bị chèn DLL và giảm DLL’s lock count đi. DLL chèn vào sẽ tự xóa khỏi không gian vùng nhớ của process.

Ưu điểm:

* Hỗ trợ Windows NT/2K và họ Windows 9x hy vọng sẽ hỗ trợ các phiên bản Windows tiếp theo.
* Chủ động gọi hàm UnhookWindowsHookEx() khi không cần thiết phải Hook nữa.

Nhược điểm:

* Có thể làm giảm đáng kể hiệu suất của hệ thống vì làm tăng xử lý cho mỗi message.
* Ảnh hưởng đến việc xử lý của toàn hệ thống và có thể phải khởi động lại để khôi phục lỗi.

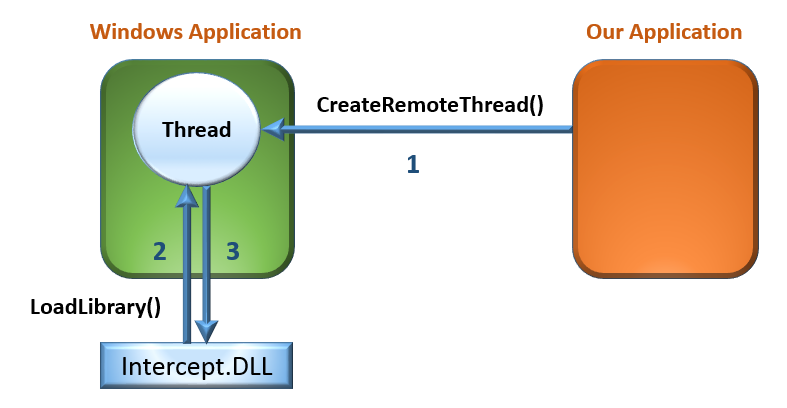
### Injecting DLL by using CreateRemoteThread() API function

* Vấn đề

Cài đặt và hook trên nhiều tiến trình không cần thiết có thể làm chậm hệ thống và gây ra những lỗi nghiêm trọng. Liệu có cách nào chỉ xử lý trên 1 số tiến trình xác định? Khi đó, chúng ta cần một giải pháp mới tập trung vào xử lý một tiến trình.

* Giải pháp

Sử dụng CreateRemoteThread() để tạo tiểu trình trong tiến trình xác định và tải DLL vào tiến trình mục tiêu.



Hình ‑ Sử dụng CreateRemoteThread() tải DLL vào ứng dụng.

Về cơ bản, mỗi tiến trình thực hiện tải DLL động bằng hàm **LoadLibrary**(). Vấn đề là làm sao để gọi hàm load DLL từ bên ngoài tiến trình? Ý tưởng là tạo một thread con trong tiến trình đó và load DLL từ bên ngoài vào trong tiến trình. Để hiện thực hóa ý tưởng này, chúng ta có hàm **CreateRemoteThread**() và **CreateRemoteThreadEx**() sẽ tạo một thread chạy trong vùng nhớ ảo của tiến trình.

HANDLE WINAPI CreateRemoteThread(

\_In\_   HANDLE hProcess,

\_In\_   LPSECURITY\_ATTRIBUTES lpThreadAttributes,

\_In\_   SIZE\_T dwStackSize,

\_In\_   LPTHREAD\_START\_ROUTINE lpStartAddress,

\_In\_   LPVOID lpParameter,

\_In\_   DWORD dwCreationFlags,

\_Out\_  LPDWORD lpThreadId

);

Các tham số cần chú ý:

* *lpStartAddress* [in]: Một con trỏ tới hàm ứng dụng định nghĩa kiểu **LPTHREAD\_START\_ROUTINE**  sẽ được thực thi bởi thread mới tạo và đóng vai trò là địa chỉ bắt đầu của thread được tạo từ xa.
* *lpParameter* [in]: Con trỏ tới một biến truyền tới hàm sẽ được thực thi bởi thread mới.

Ví dụ: chương trình tiến hành chèn **Hook.dll** vào ứng dụng có handle là hProcessForHooking với tham số "C:\\Hook.dll"

hThread = ::CreateRemoteThread(

hProcessForHooking,

NULL,

0,

pfnLoadLibrary,

"C:\\Hook.dll",

0,

NULL);

* Trong đó **pfnLoadLibrary** là con trỏ hàm của **LoadLibrary()**. Sử dụng hàm **GetProcAddress()** để lấy địa chỉ hàm này. May mắn là trong Kernel32.DLL luôn ánh xạ địa chỉ giống nhau ở tất cả process nên địa chỉ hàm lấy được luôn hợp lệ.
* Tham số ngay sau đó là đường dẫn tới DLL cần load: "C:\\Hook.dll"

Một lưu ý quan trọng khi sử dụng phương pháp này đó là truyền cờ **PROCESS\_ALL\_ACCESS** khi **OpenProcess().** Điều này giúp ta có đầy đủ quyền truy cập trong process.

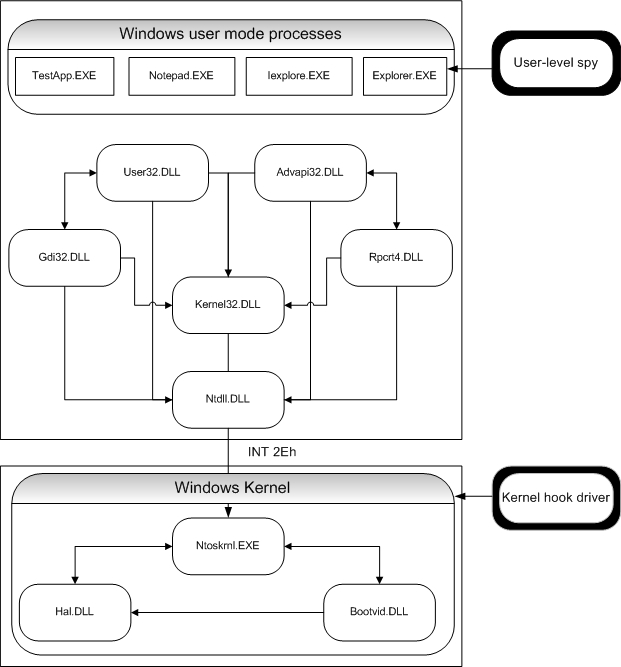
## Cơ chế chặn hàm API

* Vấn đề

Giả sử chúng ta đã chèn được DLL vào process cần xử lý. Vấn đề tiếp theo là làm thế nào để biết được hàm API xảy ra khi nào và tiến hành thay đổi xử lý bên trong hàm API.

* Giải pháp

Trước hết, chúng ta cần hiểu là Windows có 2 chế độ là User mode và Kernel mode. User mode là chế độ mà hệ điều hành cho phép chúng ta dễ dàng can thiệp và xử lý. Các thao tác của người dùng sẽ được các phương thức ở mức User mode xử lý và gọi xuống các hàm ở Kernel. Kernel mode đòi hỏi lập trình viên phải viết và xử lý trên Driver. Đối với hệ điều hành 64 bit thì Driver phải được Microsoft ký mới được thực thi các chức năng trong đó.



Hình ‑ User mode và Kernel mode trong Windows [14]

Chúng em chủ yếu tìm hiểu về Win32 User Level hooking.

### Windows subclassing

Sử dụng phương thức **SetWindowLongPtr**() với tham số **GWLP\_WNDPROC** để truyền con trỏ tới windows procedure mới. Khi đã cài đặt một subclass, mỗi khi có message tới windows được chỉ định (A) nó sẽ tìm đến địa chỉ của windows procedure liên kết với windows tương ứng (A) và gọi procedure của bạn thay vì hàm gốc.

Giới hạn: Chỉ hợp lệ với các process đặc biệt và ứng dụng không nên có subclass của windows tạo bởi process khác.

Thường sử dụng hook thông qua add-in và có thể lấy được handle của window mà bạn muốn thay thế procedure của nó.

### Proxy DLL

Thay thế một DLL cùng tên và export tất cả các tính năng của DLL gốc. Kỹ thuật này dễ dàng cài đặt bằng function forwarders. Sử dụng #pragma comment :

#pragma comment(linker, "/export:DoSomething=DllImpl.ActuallyDoSomething")

### Code overwriting

Đây là phương pháp hook rất phổ biến được áp dụng như là một trong những phương pháp đầu tiên để chặn hàm API.



Hình ‑ Quá trình chặn hàm API.

Bước đầu tiên, chúng ta phải xác định chính xác địa chỉ hàm API trong bộ nhớ. Sử dụng hàm GetProcessAddress() để lấy địa chỉ hàm cần hook.

Lưu địa chỉ hàm gốc lại để có thể khôi phục lại hoặc sử dụng trong quá trình hook.

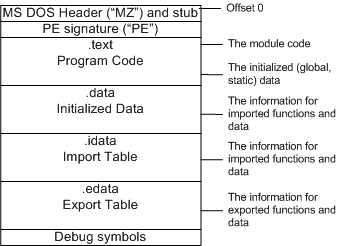
Ghi đè một lệnh JMP vào địa chỉ hàm gốc để chuyển hướng đến hàm tự định nghĩa (hook function).

Trong hàm tự định nghĩa, chúng ta toàn quyền xử lý dữ liệu tham số đã hook được.

Gọi lại hàm gốc ban đầu nếu cần thiết và khôi phục lại địa chỉ hàm gốc khi hoàn thành hook.

### Spying by altering of the Import Address Table

PE format là cấu trúc dữ liệu đóng gói thông tin cần thiết cho Windows OS loader quản lý mã thực thi [15]. Nhưng thông tin này bao gồm dữ liệu DLL, API export và import table, thông tin quản lý tài nguyên và thread-local-storage (TLS). Trên hệ điều hành NT, PE format sử dụng cho EXE, DLL, SYS (device driver) và các kiểu file khác [15].



Hình ‑ PE format khi thực thi chương trình [14]

Trong đó, Import Address Table (IAT nằm trong phần .idata) sẽ lưu địa chỉ hàm của module binary (DLL) khác. Mỗi khi ứng dụng cần gọi hàm sẽ tìm trong bảng này. Như thế sẽ tiết kiệm hơn khi ứng dụng cần sử dụng các hàm import.

Thay đổi địa chỉ hàm trong IAT sẽ giúp chuyển hướng tới hàm hook của mình. Đây chính là cách can thiệp hàm API bằng phương pháp này.

## Microsoft Office Add-in

Microsoft cho phép lập trình viên viết những Add-in tích họp cùng với Office. Một ứng ụng Add-in này có thể bắt được các sự kiện của một file Office từ khi mở lên đọc dữ liệu đến khi file được đóng lại.

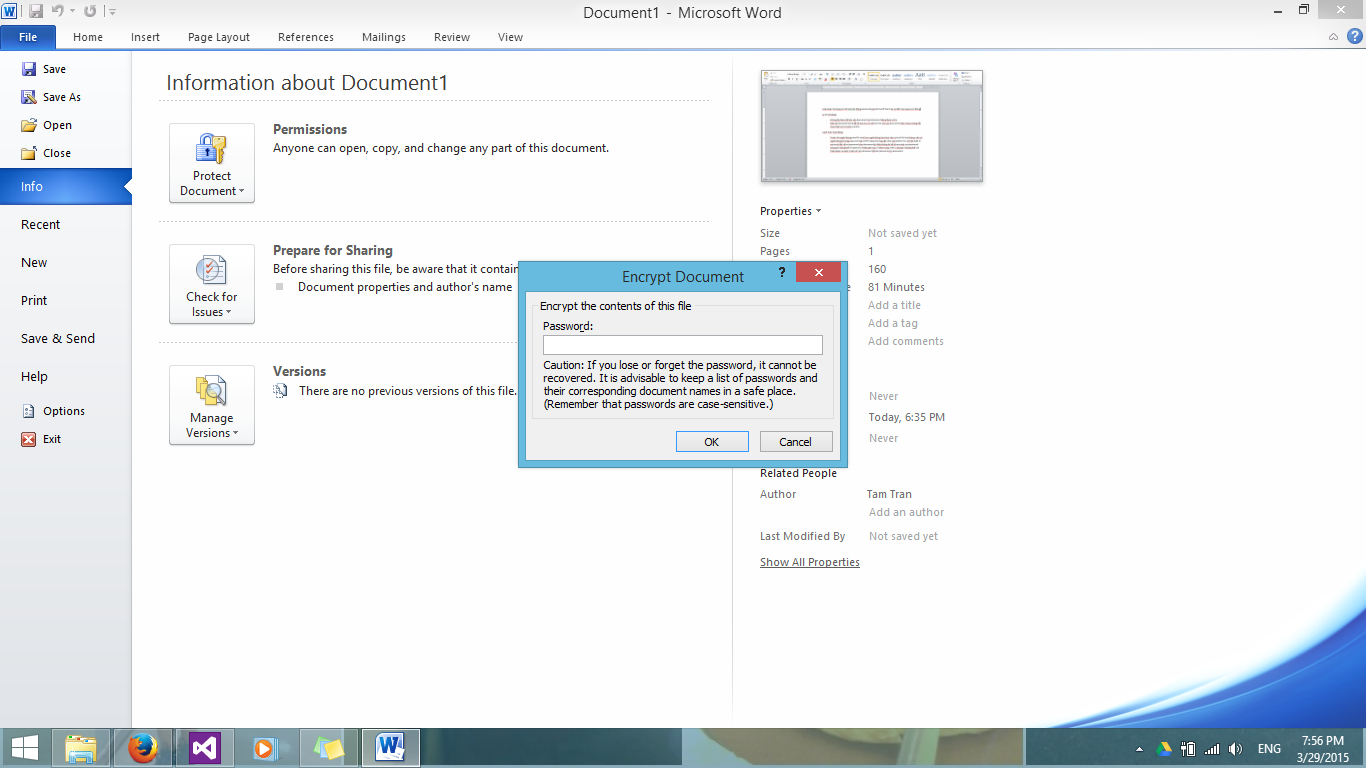
### Sử dụng cơ chế mã hóa mặc định của Word

* + Vấn đề

Dữ liệu văn phòng thường được soạn thảo bằng Microsoft Word (hay Office nói chung). Vấn đề đặt ra dữ liệu trong file Word cần được bảo vệ.

* + Giải pháp:

Sử dụng cơ chế lock file bằng password của Microsoft Word và thể hiện nó một cách trực quan sinh động hơn cho người dùng.



Hình ‑ Mã hóa dữ liệu của Microsoft Word

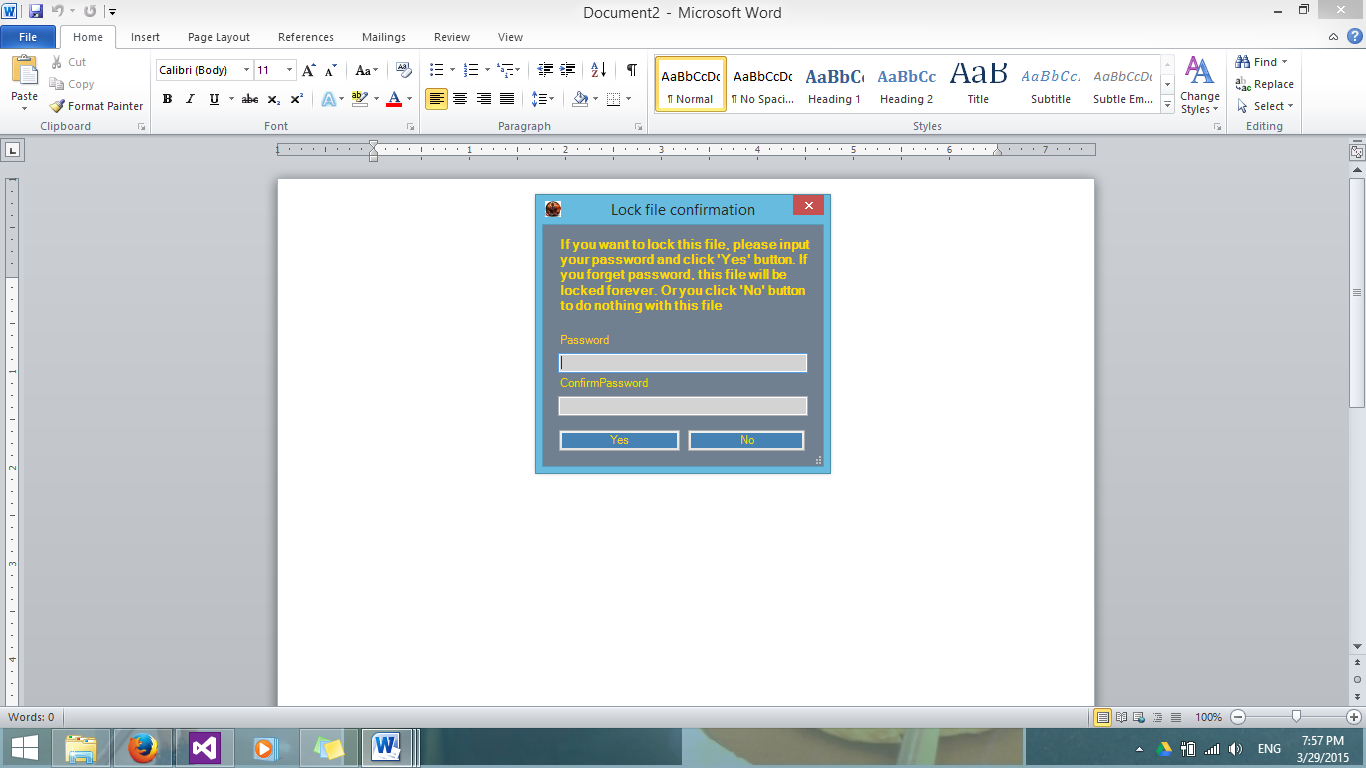
**Lý do sử dụng:**

Không lấy được dữ liệu của document khi document đang được xử lý.

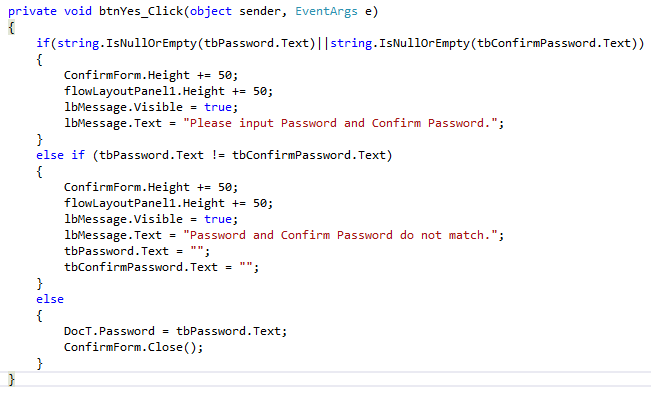
Mặc dù Microsoft Word đã hỗ trợ cho cơ chế lock file cho các document khác nhau nhưng vẫn chưa thật sự tự nhiên (smooth).

**Cách thức hoạt động:**

Trước khi người dùng save file ta sẽ cho người dùng lựa chọn việc có lock file hay không, nếu có người dùng sẽ nhập **password** và **confirm password** sau đó click vào button Yes. Khi đó Add-In mà ta cài đặt sẽ set password cho document ấy. Nếu không thì sẽ bỏ qua việc set password.



Hình ‑ Giao diện mới khi lock file

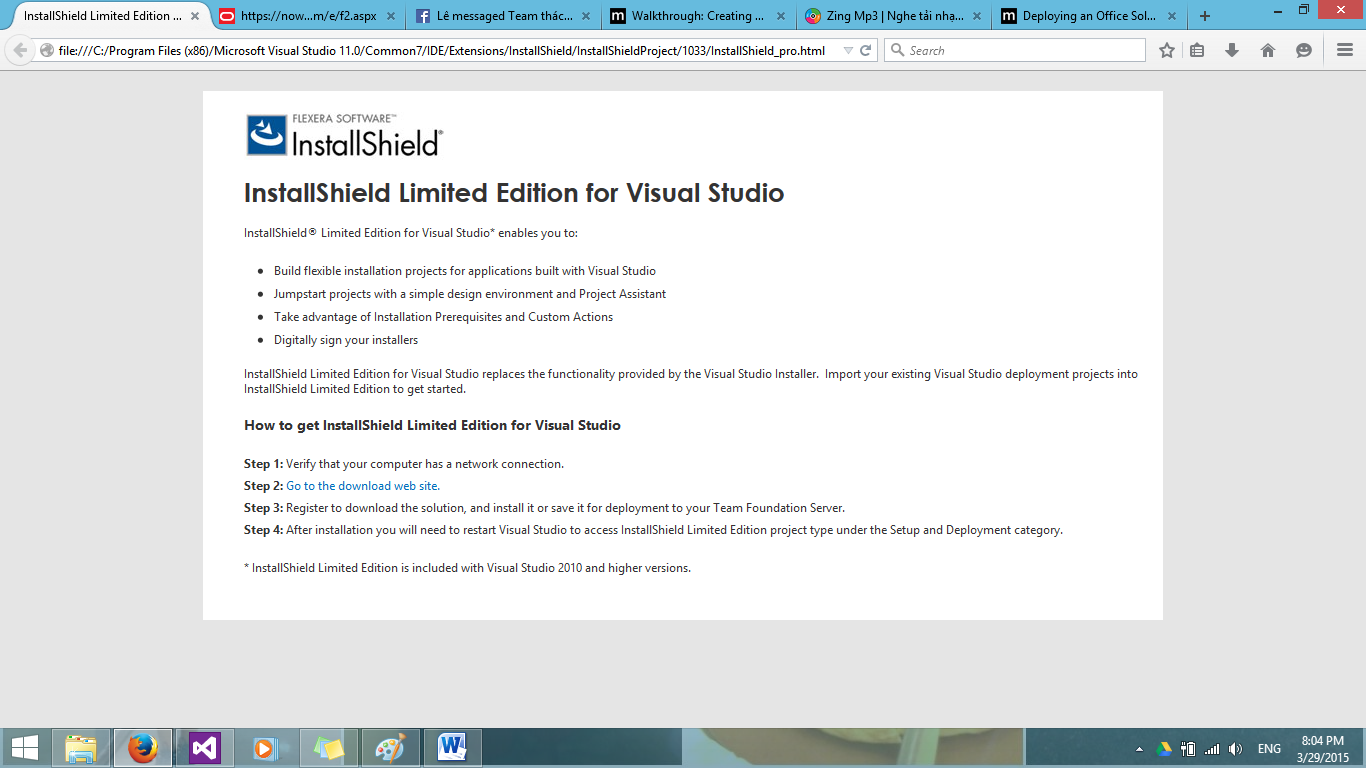


Chúng ta không thể set password một cách tùy ý ở bên trong code vì chúng ta không thể can thiệp được sự kiện trước khi mở document để set password cho document.

**Cài đặt:**

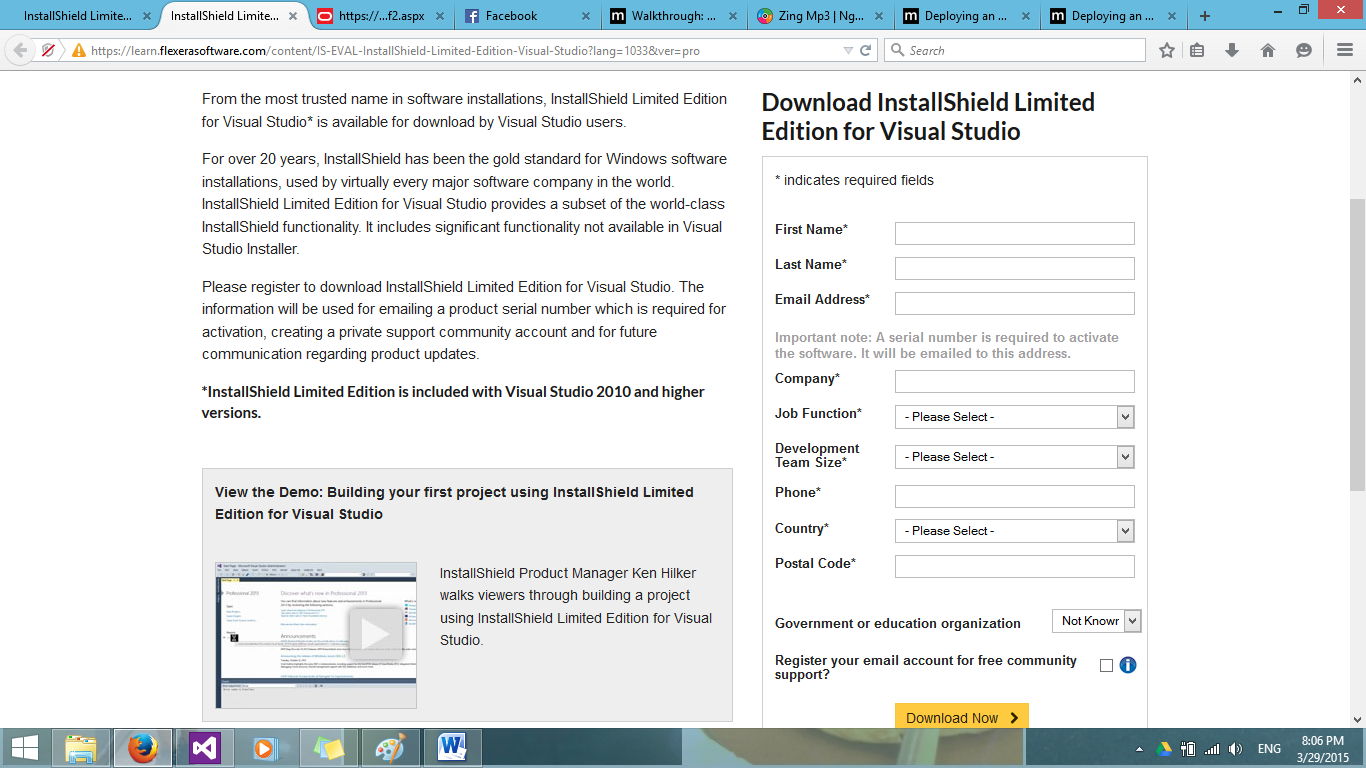
Hiện tại sẽ sử dụng Windows Installer. Làm theo các bước được hướng dẫn theo link sau: <https://msdn.microsoft.com/en-us/library/cc442767%28v=vs.110%29.aspx>

Nhưng tồn tại vấn đề sau: Khi tạo project **Enable InstallShield Limited Edition** thì sẽ hiện ra một link sau



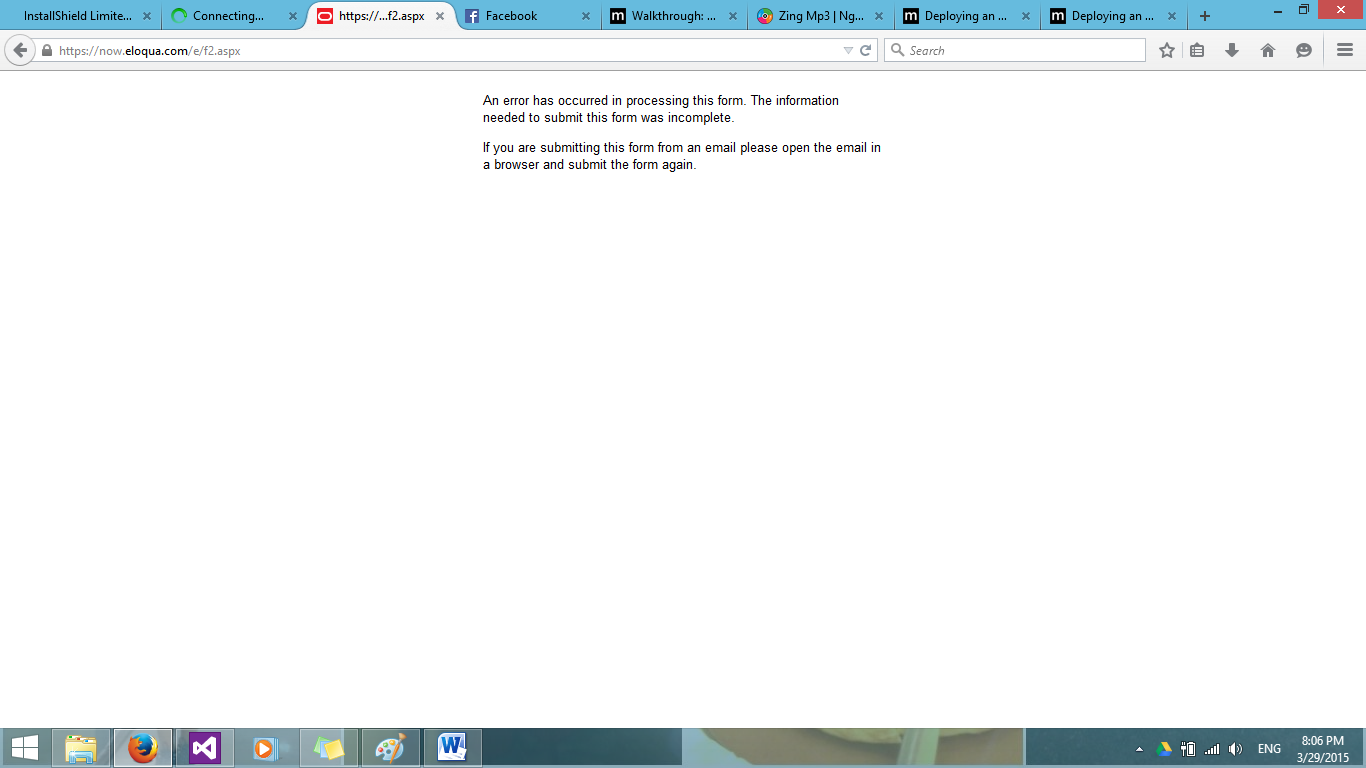
Hình ‑ InstallShield Limited Edition

Click theo đường dẫn để đăng ký để được download nhưng bước đăng ký vẫn không hoàn thành được mặc dù đã thử nhiều lần.



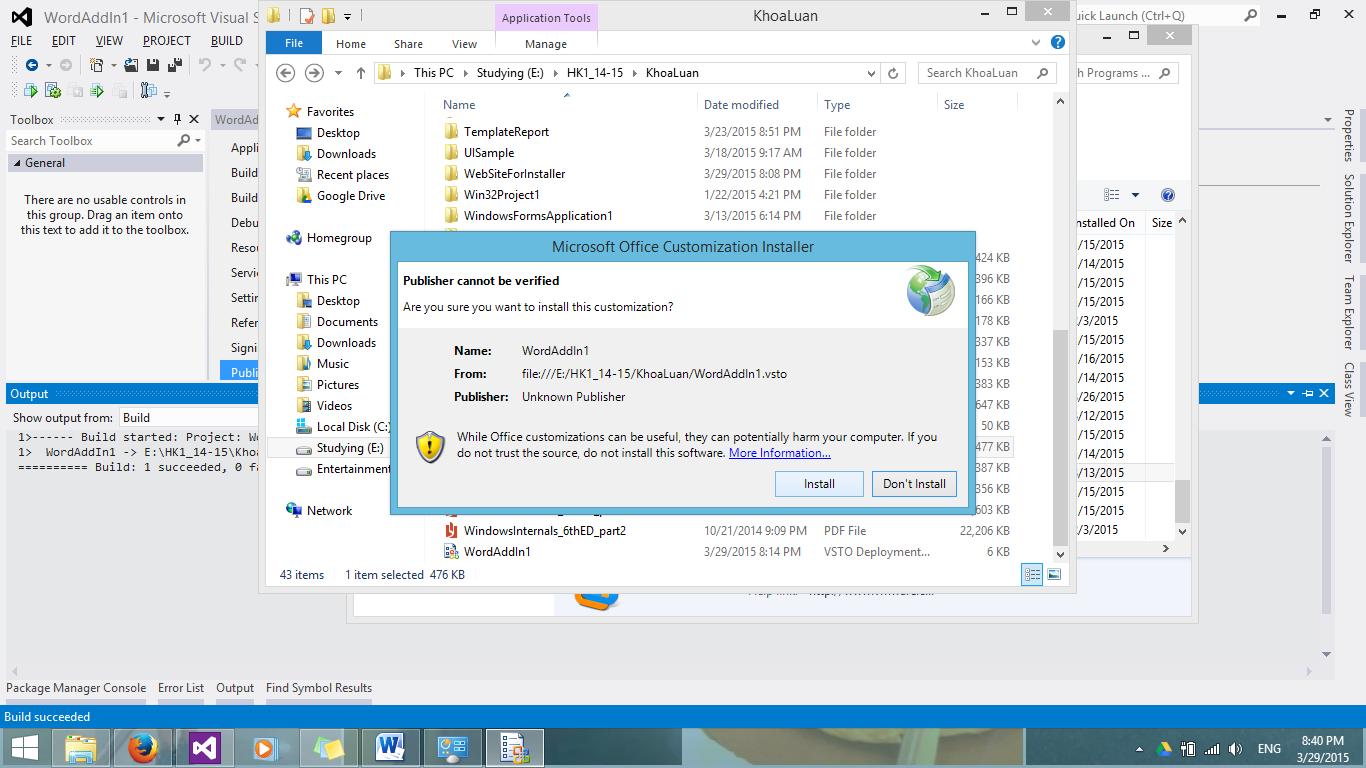
Hình ‑ InstallShield Limited Edition Download

Lỗi.

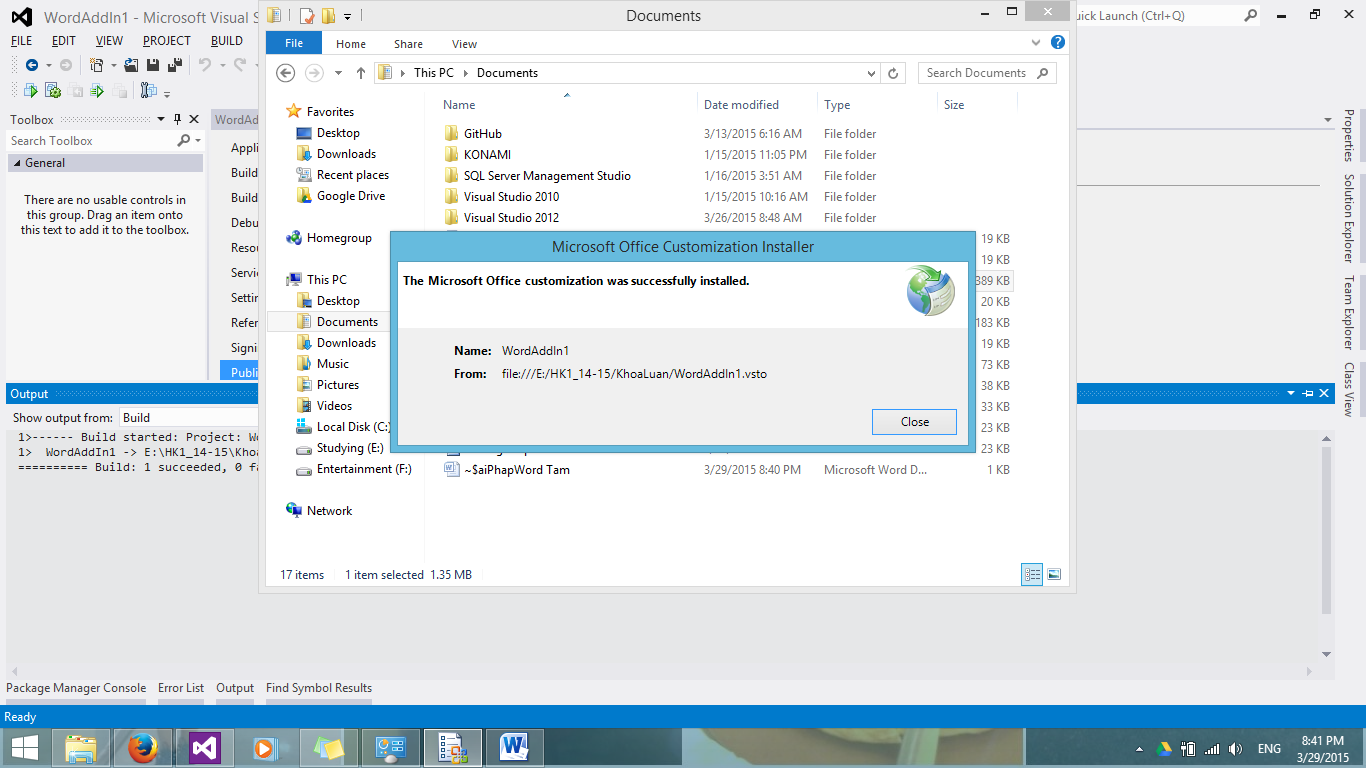


Hình ‑ InstallShield Limited Edition Error

Về cách sử dụng ClickOne <https://msdn.microsoft.com/en-us/library/bb772100%28v=vs.110%29.aspx> theo đường dẫn vừa nêu thì thành công và được kết quả sau nhưng hiện tại chỉ chạy được local: ?



Hình ‑ Cài đặt WordAddin1

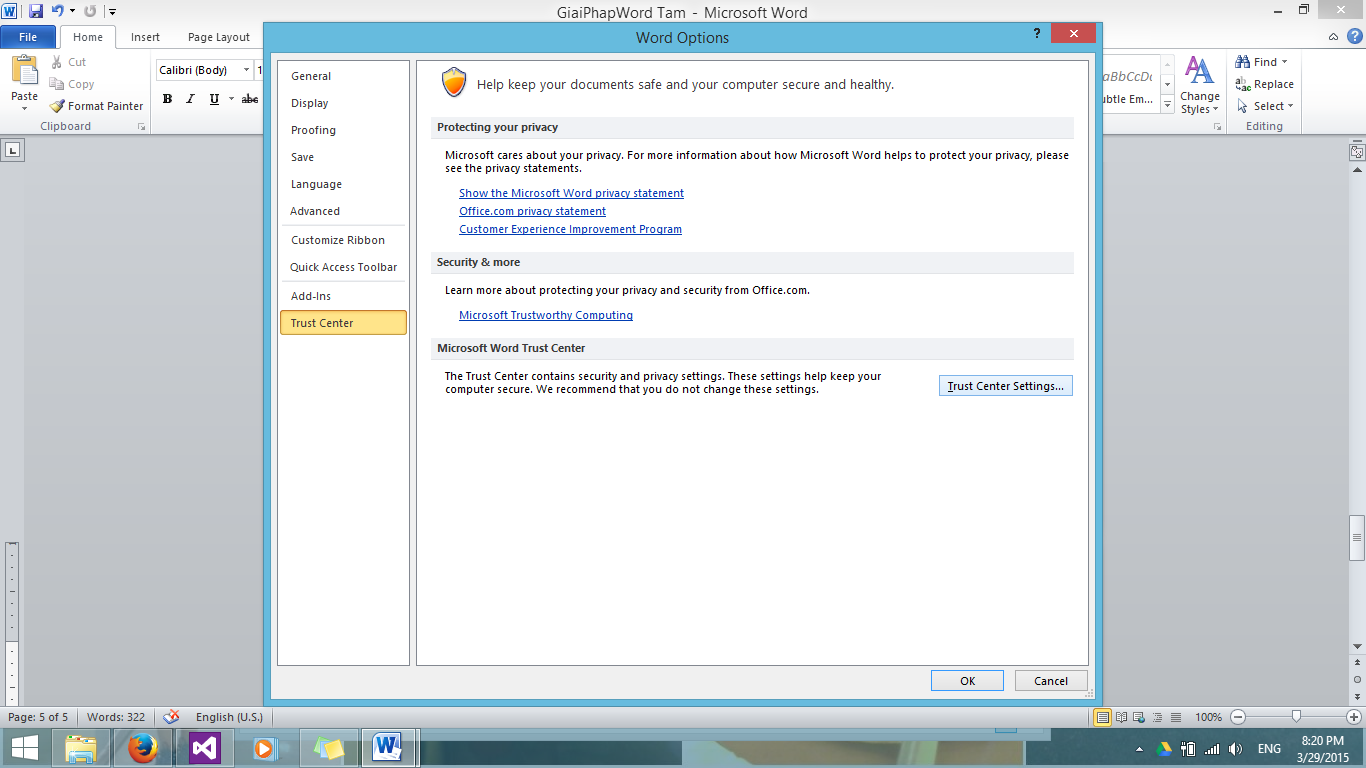


Hình ‑ Cài đặt WordAddin1

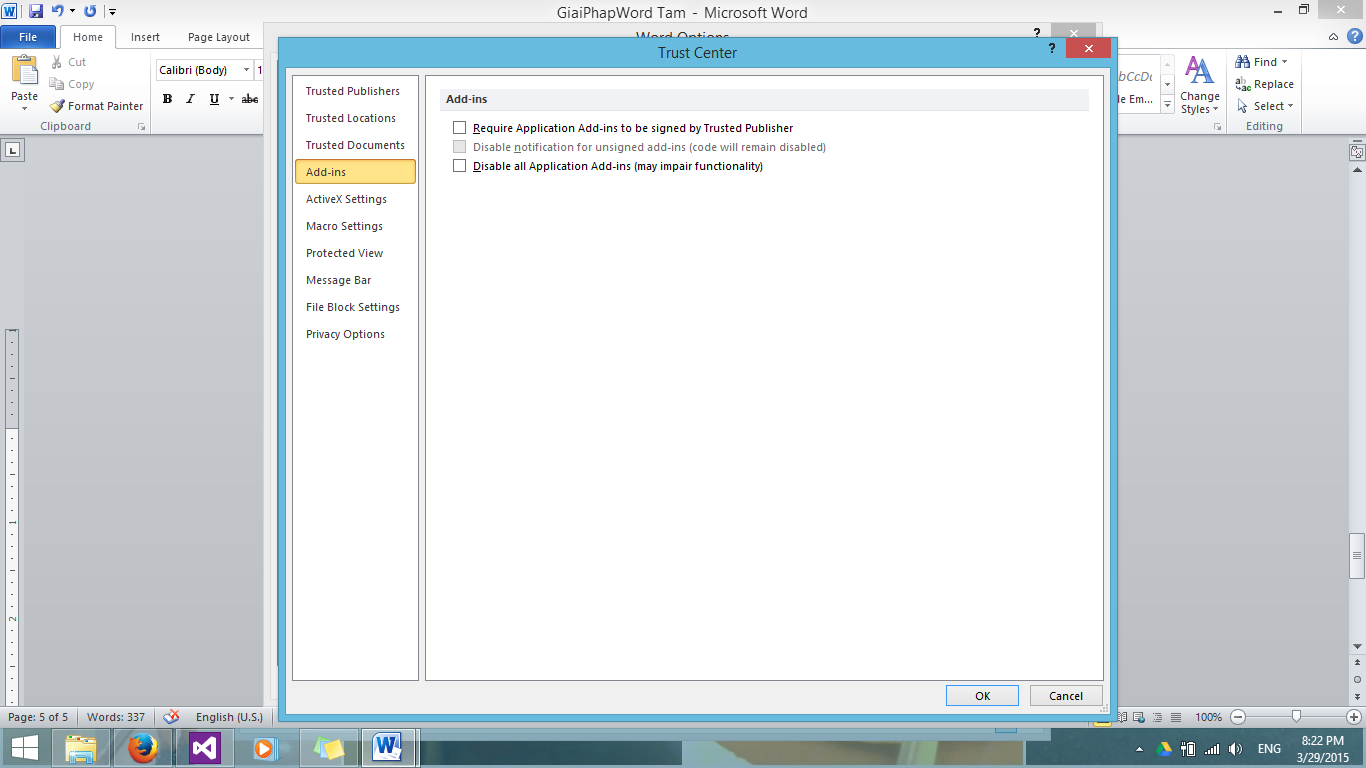
Với các vấn đề về security :

Về việc cài đặt thì vẫn chưa tìm hiểu được do bước cài đặt chưa được xử lý

Về các điều kiện để thực thi thì Microsoft Word có phần gọi là Trust Center quản lý các mức độ tin tưởng ví dụ như yêu cầu add-in phải có chữ ký của Trusted Publisher hoặc tắt hết các add-ins. Hầu như ai cũng có thể vào thay đổi settings nếu họ biết.

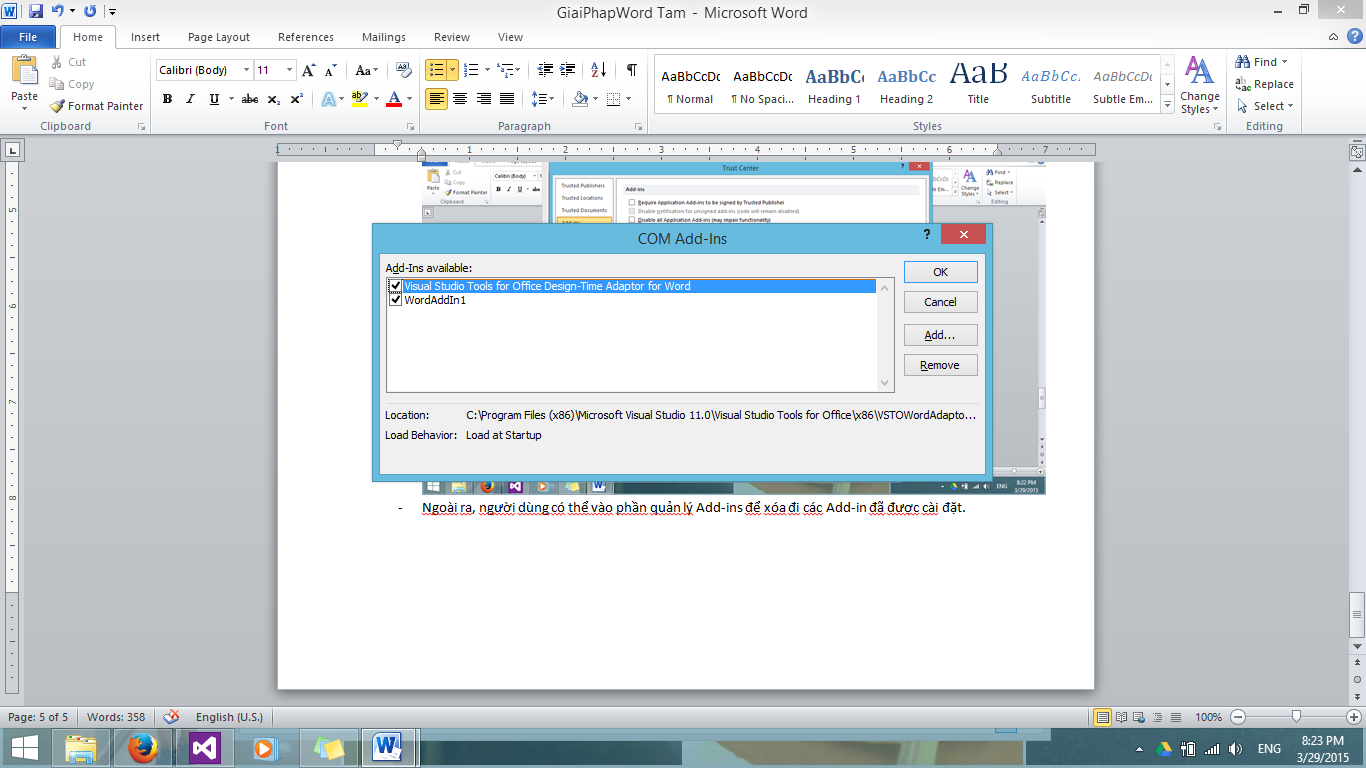


Hình ‑ Word Options – Trust Center



Hình ‑ Trust Center – Add-ins

Ngoài ra, người dùng có thể vào phần quản lý Add-ins để xóa đi các Add-in đã được cài đặt.



Hình ‑ Quản lý Add-ins trong Word

### Tự mã hóa dữ liệu

* Vấn đề

Sử dụng cơ chế mã hóa do Microsoft Office cung cấp đơn giản nhưng không kiểm soát được dữ liệu. Hơn thế nữa, thao tác mã hóa này không tự nhiên khi người dùng phải đặt mật khẩu cho từng file dữ liệu. Yếu tố này đặt ra vấn đề tự mã hóa dữ liệu trong cài đặt Add-in.

* Giải pháp

Đối với vấn đề này, chúng em đưa ra 2 giải pháp thử nghiệm.

Giải pháp 1: Mã hóa nội dung file theo dạng text.

Sử dụng thuật toán mã hóa TRIPDES mã hóa nội dung file dạng text. Vấn đề gặp phải là nội dung sau khi giải mã bị mất format, mất hình ảnh, biểu tượng... Vì ứng dụng Add-in không thể lấy toàn bộ nội dung file office dạng bytes nên không bảo toàn được định dạng.

Giải pháp 2: Mã hóa toàn bộ file trước khi đóng ứng dụng.

Chúng em thử bắt sự kiện trước khi chương trình đóng ứng dụng, sau đó đọc toàn bộ dữ liệu của file để mã hóa. Nhưng giải pháp này cũng không khả quan khi chương trình không được phép truy xuất dữ liệu khi file vẫn đang mở bằng Word.

## Phát hiện thiết bị lưu trữ tương tác với máy tính

* Vấn đề

Thiết bị lưu trữ di động cần được bảo vệ dữ liệu trong đó. Trước hết, chúng ta cần biết khi nào thiết bị tương tác với máy tính để tiến hành xử lý dữ liệu được đưa vào.

* Giải pháp

Chúng ta có 2 giải pháp C++ và C# sau đây:

### Giải pháp C++

#### Lý thuyết

Trong Windows, mỗi ứng dụng sẽ gửi thông báo sự kiện bằng hàm **BroadcastSystemMessage()**. Mọi ứng dụng tầng trên có thể nhận thông báo bằng cách xử lý tin nhắn WM\_DEVICECHANGE. Ứng dụng có thể dùng hàm **RegisterDeviceNotification()** để đăng ký nhận thông báo về các thiết bị.

Service có thể dùng hàm **RegisterDeviceNotification()** để đăng ký nhận thông báo thiết bị. Nếu một service chỉ định một window handle trong **hRecipient** parameter, thông báo được gửi đến window procedure. Nếu hRecipient là một dịch vụ kiểm soát trạng thái, **SERVICE\_CONTROL\_DEVICEEVENT** thông báo được gửi đến cho dịch vụ điều khiển xử lý.

Sự kiện **DBT\_DEVICEARRIVAL** và **DBT\_DEVICEREMOVECOMPLETE** được tự động phát thông báo cho tất cả các ứng dụng tầng trên cho cổng các thiết bị. Hơn thế nữa, việc này không cần thiết gọi RegisterDeviceNotification cho các cổng, và hàm thất bại nếu dbch\_devicetype là **DBT\_DEVTYP\_VOLUME**. Thiết bị OEM-defined không sử dụng trực tiếp bởi hệ thống, do đó cá chức năng thất bại nếu dbch\_devicetype là **DBT\_DEVTYP\_OEM**.

Khai báo **GUID** cho tất cả các **USB serial host PnP drivers** (có thể thay thế bằng bất kỳ GUID hợp lệ khác)

// GUID sử dụng cho tất cả USB serial host PnP drives, nhưng

// có thể thay thế bằng những device class guid hợp lệ khác

GUID WceusbshGUID = { 0x25dbce51, 0x6c8f, 0x4a72,

0x8a, 0x6d, 0xb5, 0x4c, 0x2b, 0x4f, 0xc8, 0x35 };

BOOL DoRegisterDeviceInterfaceToHwnd(

IN GUID InterfaceClassGuid,

IN HWND hWnd,

OUT HDEVNOTIFY \*hDeviceNotify

)

{

DEV\_BROADCAST\_DEVICEINTERFACE NotificationFilter;

ZeroMemory(&NotificationFilter, sizeof(NotificationFilter));

NotificationFilter.dbcc\_size = sizeof(DEV\_BROADCAST\_DEVICEINTERFACE);

NotificationFilter.dbcc\_devicetype = DBT\_DEVTYP\_DEVICEINTERFACE;

NotificationFilter.dbcc\_classguid = InterfaceClassGuid;

\*hDeviceNotify = RegisterDeviceNotification(

hWnd, // events recipient

&NotificationFilter, // type of device

DEVICE\_NOTIFY\_WINDOW\_HANDLE // type of recipient handle

);

if (NULL == \*hDeviceNotify)

{

ErrorHandler(TEXT("RegisterDeviceNotification"));

return FALSE;

}

return TRUE;

}

Hàm **DoRegisterDeviceInterfaceToHwnd**(IN GUID InterfaceClassGuid, IN HWND hWnd, OUT HDEVNOTIFY \*hDeviceNotify) có chức năng đăng ký cho một Hwnd để nhận thông báo thay đổi giao tiếp của các thiết bị có GUID được chỉ định ở **WcenusbshGUID**. Bên trong hàm **DoRegisterDeviceInterfaceToHwnd**() gọi hàm:

HDEVNOTIFY WINAPI RegisterDeviceNotification(

\_In\_ HANDLE hRecipient,

\_In\_ LPVOID NotificationFilter,

\_In\_ DWORD Flags

);

hRecipient [in]: một handle của window hoặc service sẽ nhận các sự kiện của thiết bị được chỉ định trong tham số *NotificationFilter*. Những window handle tương tự có thể sử dụng trong multiple calls tới **RegisterDeviceNotification**. Service có thể xác định cả windows handle và service status handle.

NotificationFilter [in]: Một con trỏ tới block dữ liệu xác định kiểu thiết bị mà thông báo nên được gửi. Block này luôn bắt đầu với cấu trúc **DEV\_BROADCAST\_HDR**. Dữ liệu theo sau header này phụ thuộc vào giá trị của **dbch\_devicetype**, nó có thể là **DBT\_DEVTYP\_HANDLE** hoặc **DBT\_DEVTYP\_DEVICEINTERFACE**.

Flags [in]: Tham số này có thể là một trong các giá trị sau:

|  |  |
| --- | --- |
| **DEVICE\_NOTIFY\_WINDOW\_HANDLE**  0x00000000 | Tham số *hRecipient* là một window handle. |
| **DEVICE\_NOTIFY\_SERVICE\_HANDLE**  0x00000001 | Tham số *hRecipient* là một service status handle. |

Hàm **WindProcCallback**(…) tiến hành đăng ký nhận thông báo trong WM\_CREATE. Gọi hàm **DoRegisterDeviceInterfaceToHwnd**() để đăng ký một lần duy nhất khi khởi tạo

INT\_PTR WINAPI WinProcCallback(

HWND hWnd,

UINT message,

WPARAM wParam,

LPARAM lParam

)

{

LRESULT lRet = 1;

static HDEVNOTIFY hDeviceNotify;

static HWND hEditWnd;

static ULONGLONG msgCount = 0;

switch (message)

{

case WM\_CREATE:

if (!DoRegisterDeviceInterfaceToHwnd(WceusbshGUID,

hWnd, &hDeviceNotify))

{

// Terminate on failure. ErrorHandler(TEXT("DoRegisterDeviceInterfaceToHwnd"));

ExitProcess(1);

}

break;

…

}

return lRet;

}

Cuối cùng, tạo cửa sổ giao diện trong WM\_CREATE

hEditWnd = CreateWindow(TEXT("EDIT"),// predefined class

NULL, // no window title

WS\_CHILD | WS\_VISIBLE | WS\_VSCROLL |

ES\_LEFT | ES\_MULTILINE | ES\_AUTOVSCROLL,

0, 0, 0, 0, // set size in WM\_SIZE message

hWnd, // parent window

(HMENU)1, // edit control ID

(HINSTANCE)GetWindowLong(hWnd, GWL\_HINSTANCE),

NULL); // pointer not needed

if (hEditWnd == NULL)

{

// Terminate on failure.

ErrorHandler(TEXT("CreateWindow: Edit Control"));

ExitProcess(1);

}

// Add text to the window.

SendMessage(hEditWnd, WM\_SETTEXT, 0,

(LPARAM)TEXT("Registered for USB device notification...\n"));

break;

Xử lý trong WM\_DEVICECHANGE, khi nhận một device mới:

case WM\_DEVICECHANGE:

{

PDEV\_BROADCAST\_DEVICEINTERFACE b = (PDEV\_BROADCAST\_DEVICEINTERFACE)lParam;

TCHAR strBuff[256];

switch (wParam)

{

case DBT\_DEVICEARRIVAL:

msgCount++;

StringCchPrintf(

strBuff, 256,

TEXT("Message %d: DBT\_DEVICEARRIVAL\n"),msgCount);

break;

Xuất ra màn hình thông báo khi thiết bị rời máy tính

case DBT\_DEVICEREMOVECOMPLETE:

msgCount++;

StringCchPrintf( strBuff, 256,TEXT("Message %d: DBT\_DEVICEREMOVECOMPLETE\n"), msgCount);

break;

case DBT\_DEVNODES\_CHANGED:

msgCount++;

StringCchPrintf( strBuff, 256,TEXT("Message %d: DBT\_DEVNODES\_CHANGED\n"), msgCount);

break;

Tương tự đối với Windows Message khác nhận được, chương trình hủy đăng ký nhận thông báo khi thoát.

case WM\_CLOSE:

if (!UnregisterDeviceNotification(hDeviceNotify))

{

ErrorHandler(TEXT("UnregisterDeviceNotification"));

}

DestroyWindow(hWnd);

break;

Khi đóng ứng dụng, thực hiện thao tác hủy đăng ký nhận thông báo với hàm

WINUSERAPI

BOOL

WINAPI

UnregisterDeviceNotification(

\_In\_ HDEVNOTIFY Handle

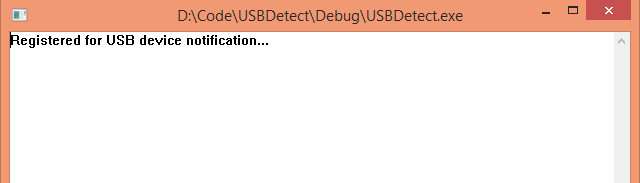
);

*Handle* [in]: được trả về khi đăng ký thành công **RegisterDeviceNotification**()

Demo

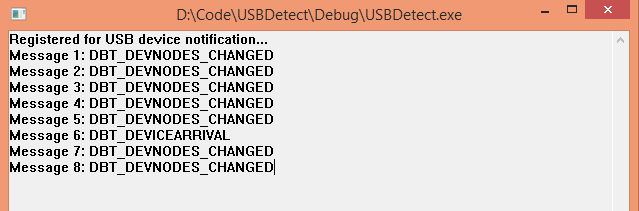
#### Demo

Giao diện khi khởi động



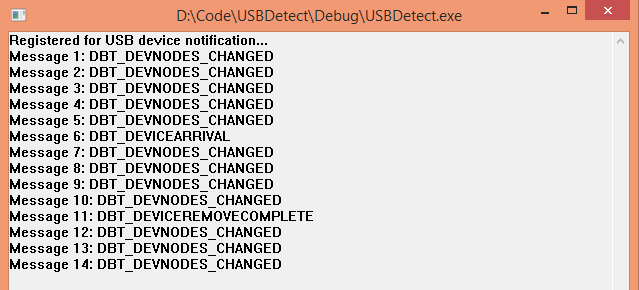
Hình ‑ Giao diện ban đầu

Khi cắm 1 USB vào máy tính



Hình ‑ Chương trình thông báo khi máy tính nhận USB

Khi rút USB ra khỏi máy tính



Hình ‑ Chương trình thông báo khi USB không tương tác

### Giải pháp C#

#### Lý thuyết

Sử dụng **WMI** để phát hiện các sự kiện **insert**/**remove** USB trên máy tinh hay laptop.

**Windows Management Instrumentation** (WMI) là việc cài đặt của Microsoft Web-Based Management Enterprise (WBEM), nó là một sáng kiến công nghiệp để phát triển một chuẩn công nghệ cho việc truy cập thông tin quản lý trong môi trường doanh nghiệp. WMI sử dụng **Common Information Model** (CIM) tiêu chuẩn công nghiệp để đại diện cho các hệ thống, các ứng dụng, mạng lưới, thiết bị, quản lý và các thành phần khác. CIM được phát triển và duy trì bởi Management Task Force phân tán (DMTF).

Khi khởi động chương trình thì ta cũng đã khởi tạo tìm kiếm 1 list chuỗi chứa các ký tự tên của các USB hiện đang có trong máy để phục vụ cho công việc xác định được USB nào bị remove.

var tmp = new ManagementObjectSearcher("select Description,Name from Win32\_LogicalDisk").Get();

foreach (var drive in tmp)

{

if (drive["Description"].ToString().Contains("Removable"))

drives.Add(drive["Name"].ToString());

}

Sau đó ta bắt đầu cho chương trình theo dõi các sự kiện InsertUSB và RemoveUSB

// Insert Device

WqlEventQuery insertQuery = new WqlEventQuery("SELECT \* FROM \_\_InstanceCreationEvent WITHIN 2 WHERE TargetInstance ISA 'Win32\_USBHub'");

ManagementEventWatcher insertWatcher =

new ManagementEventWatcher(insertQuery);

insertWatcher.EventArrived +=

new EventArrivedEventHandler(DeviceInsertedEvent);

insertWatcher.Start();

// Remove Device

WqlEventQuery removeQuery = new WqlEventQuery("SELECT \* FROM \_\_InstanceDeletionEvent WITHIN 2 WHERE TargetInstance ISA 'Win32\_USBHub'");

ManagementEventWatcher removeWatcher =

new ManagementEventWatcher(removeQuery);

removeWatcher.EventArrived +=

new EventArrivedEventHandler(DeviceRemovedEvent);

removeWatcher.Start();

Khi các sự kiện xảy ra thì các phương thức tương ứng sẽ được thực hiện. Với sự kiện insert thì ta sẽ có phương **DeviceInsertedEvent**

private static void DeviceInsertedEvent(object sender, EventArrivedEventArgs e)

{

ManagementBaseObject instance =

(ManagementBaseObject)e.NewEvent["TargetInstance"];

Console.Write("Insert - ");

string name = GetDriveLetter(instance);

drives.Add(name);

Console.WriteLine(name);

}

Ở trong phương **DeviceInsertedEvent** ta gọi đến phương **GetDriveLetter** để lấy được ký tự của USB khi được insert vào máy và thêm ký tự đó vào list chuỗi mà ta đã khởi tạo lúc đầu. Ta cũng sử dụng các class phương thức, SQL do WMI cung cấp để tìm kiếm ký tự được cấp cho USB.

public static string GetDriveLetter(ManagementBaseObject instance)

{

var tmp = new ManagementObjectSearcher("select \* from Win32\_DiskDrive where InterfaceType='USB'").Get();

var deviceid = instance["DeviceID"].ToString();

foreach (var drive in tmp)

{

if(drive["PNPDeviceID"].ToString().Contains(deviceid.Remove(0,deviceid.LastIndexOf("\\")+1)))

{

ManagementObject partition =

New ManagementObjectSearcher(String.Format("associators of {{Win32\_DiskDrive.DeviceID='{0}'}} where AssocClass = Win32\_DiskDriveToDiskPartition", drive["DeviceID"])).First();

if (partition != null)

{

// associate partitions with logical disks (drive letter volumes)

ManagementObject logical = new ManagementObjectSearcher(String.Format("associators of {{Win32\_DiskPartition.DeviceID='{0}'}} where AssocClass= Win32\_LogicalDiskToPartition",

partition["DeviceID"])).First();

if (logical != null)

{

// finally find the logical disk entry

return logical["Name"].ToString();

}

}

}

}

return "NotFound";

}

Tương tự cho sự kiện Remove USB ta có phương thức **DeviceRemovedEvent**

private static void DeviceRemovedEvent(object sender, EventArrivedEventArgs e)

{

ManagementBaseObject instance =

(ManagementBaseObject)e.NewEvent["TargetInstance"];

Console.Write("Remove - ");

var tmpQuery = new ManagementObjectSearcher("select Description,Name from Win32\_LogicalDisk").Get();

var currDrives = new List<string>();

foreach (var drive in tmpQuery)

{

If (drive["Description"].ToString().Contains("Removable"))

{

currDrives.Add(drive["Name"].ToString());

}

}

for (int i = 0; i < drives.Count; i++)

{

if (currDrives.IndexOf(drives[i]) == -1)

{

Console.WriteLine(drives[i]);

drives.Clear();

drives.AddRange(currDrives);

break;

}

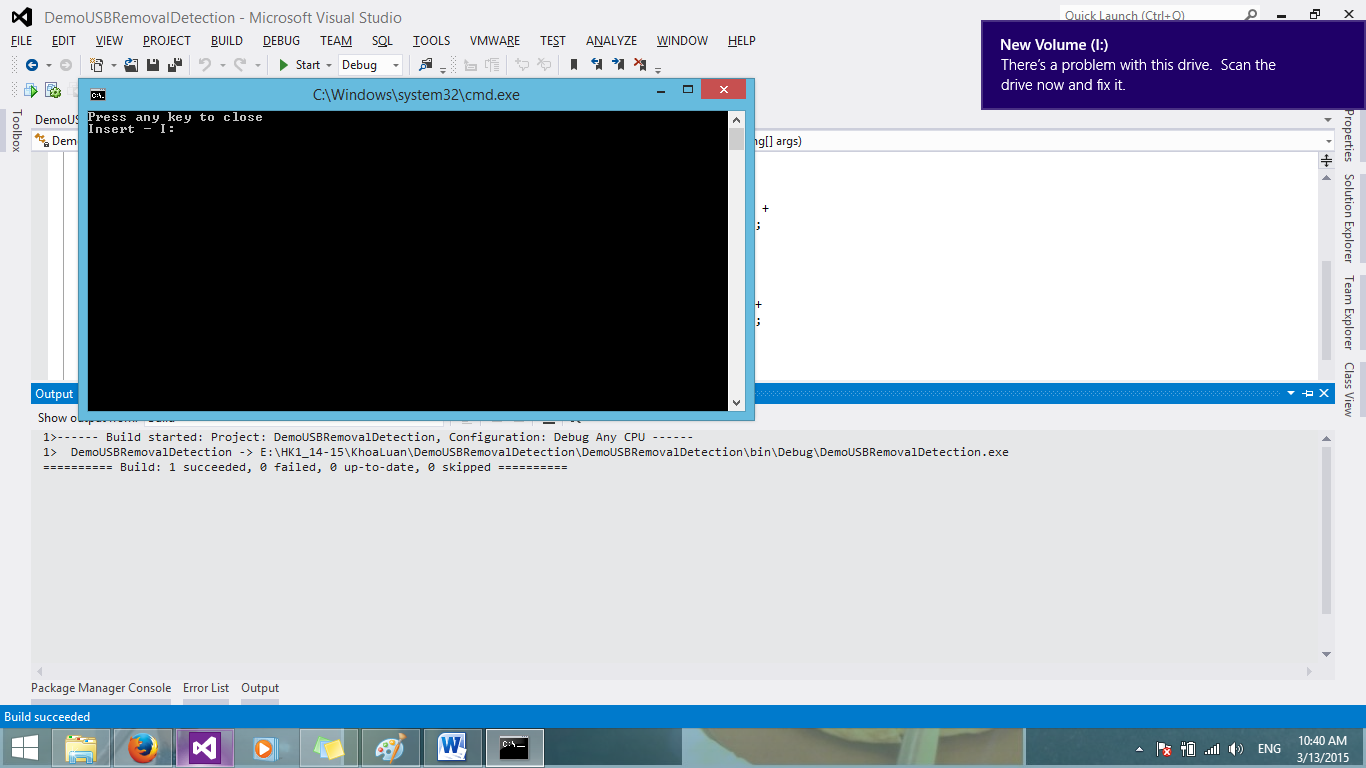
}

}

Trong đoạn code xử lý khi remove thiết bị thì điều mà ta muốn làm chính là xác định được ký tự được cấp cho USB đã remove là gì.

#### Demo

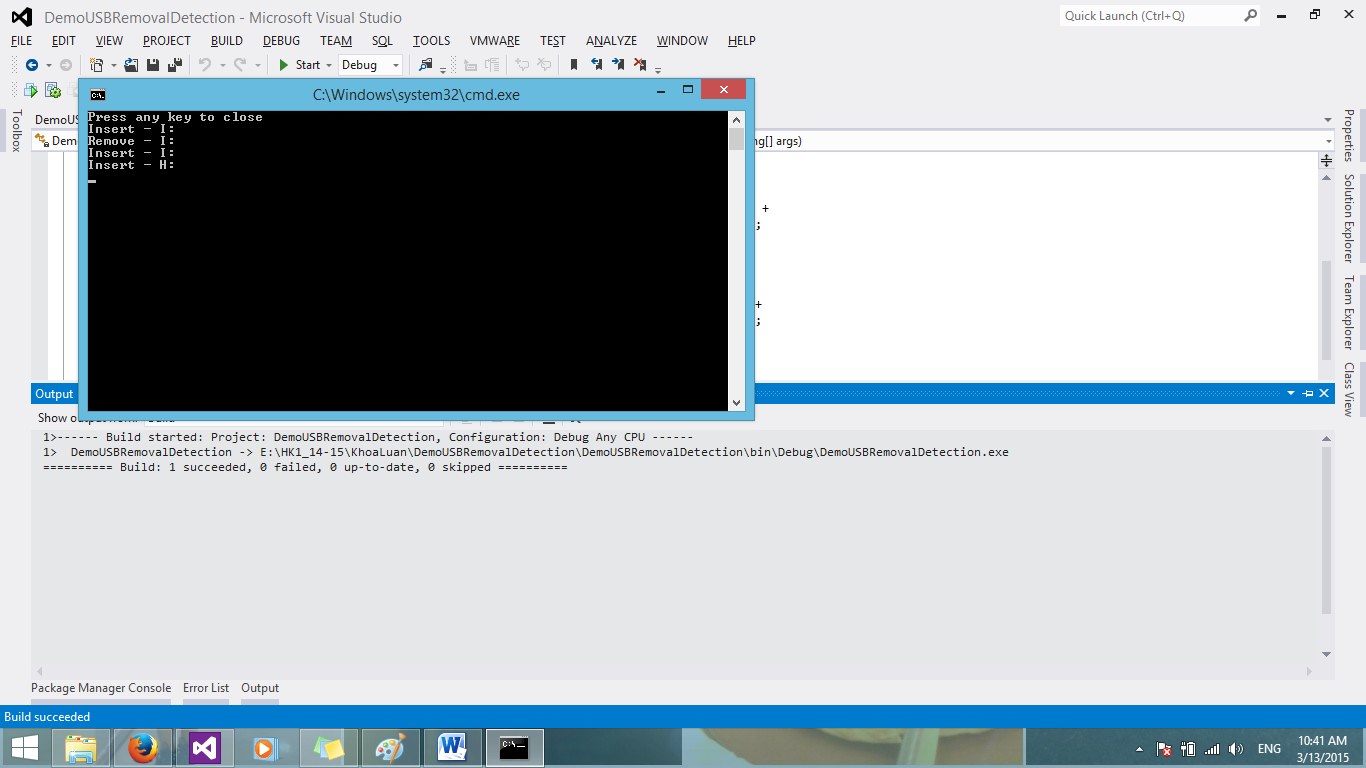
Dưới đây là hình ảnh Demo



Hình ‑ Chương trình thông báo khi có USB tương tác



Hình ‑ Chương trình thông báo khi USB không còn tương tác



Hình ‑ Thay thế một USB khác

* Vấn đề

Trong một số trường hợp, WMI không lấy được tên chính xác của device. Lúc này chương trình chỉ nhận tên device là “Not Found”. Cần có biện pháp lấy chính xác tên device có tính ổn định cao

* Giải pháp

Sử dụng class **DriveInfo** do .NET framework cung cấp để lấy được tên của thiết bị vừa kết nối vào máy tính và cũng như lấy được tên thiết bị vừa ngắt kết nối với máy tính. Ta gọi phương thức **GetDrives**() và dùng **LinQ** để lọc ra các thiết bị bộ nhớ bên ngoài máy tính. Đối với trường hợp kết nối thiết bị, ta sẽ kiểm tra xem các thiết bị nào vừa lấy được từ phương thức **GetDrives**() không có trong danh sách các thiết bị hiện tại sau đó sẽ thêm vào danh sách này. Ngược lại với trường hợp ngắt kết nối thiết bị, ta sẽ kiểm tra thiết bị nào trong danh sách hiện tại mà không tồn tại trong danh sách vừa lấy được từ phương thức **GetDrives**() thì loại ra khỏi danh sách thiết bị hiện tại.

DriveInfo[] drives = DriveInfo.GetDrives().Where(x => x.DriveType == DriveType.Removable).ToArray();

foreach (DriveInfo d in drives)

{

Console.WriteLine("Drive name: " + d.Name);

Console.WriteLine("Drive VolumeLabel:" + d.VolumeLabel);

}

# Bảo vệ dữ liệu

* Nội dung của Chương 4 trình bày giải pháp bảo vệ dữ liệu khi lưu xuống máy tính hay các thiết bị lưu trữ. Giải pháp đảm bảo an toàn và có cơ chế chia sẻ dữ liệu.

## Giải pháp bảo vệ dữ liệu

* Vấn đề

Dữ liệu trước khi lưu xuống máy tính hoặc thiết bị lưu trữ cần phải được mã hóa. Vấn đề đặt ra là chúng ta cần một cơ chết mã hóa đủ bảo mật mà không ảnh hưởng quá nhiều đến hiệu suất làm việc của máy tính.

* Giải pháp

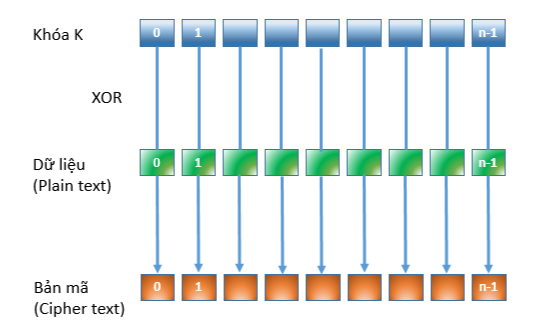
Đề xuất một cơ chế mã hóa để bảo mật dữ liệu:

### Chế độ mã hóa (Mode cipher)

Áp dụng tư tưởng stream cipher. Tức là mã hóa từng byte (hoặc bit) của dữ liệu với byte tương ứng của khóa. Ưu điểm lớn nhất của stream cipher là tốc độ mã hóa. Bởi thao tác mã hóa chỉ đơn giản là thực hiện 1 phép XOR.

Hình 4‑1 trình bày một quá trình mã hóa theo tư tưởng stream cipher điển hình. Từng phần tử (dạng bit, byte…) của dữ liệu ban đầu được XOR với từng phần từ của khóa K để tạo thành dữ liệu được mã hóa.

Quá trình này đòi hỏi chiều dài khóa K phải không nhỏ hơn chiều dài dữ liệu cần mã. Nhưng đối với những dữ liệu lớn, việc phát sinh một khóa K có chiều dài tương đương với dữ liệu ban đầu là không hiệu quả, thậm chí là không khả thi. Vì vậy, chúng ta chỉ cần phát sinh một khóa K có độ dài đảm bảo độ an toàn và ghép lại để có khóa K có độ dài bất như mong muốn. Thực tế, dữ liệu sẽ được chia nhỏ thành nhiều block có độ dài bằng khóa K (block cuối cùng có thể có độ dài ngắn hơn) và tiến hành mã hóa hoặc giải mã. Vấn đề này sẽ được trình bày rõ hơn ở phần sau.



Hình ‑ Cơ chế mã hóa Stream Cipher.

### Mã hóa và giải mã

#### Quá trình mã hóa

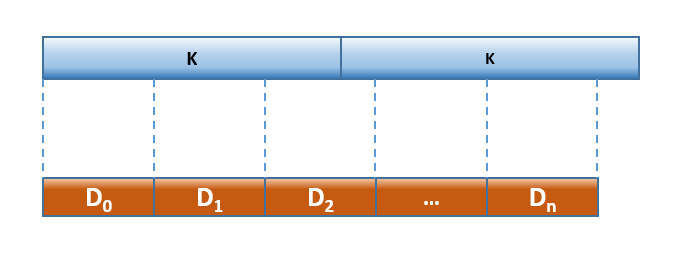
* Vấn đề

Mã hóa stream cipher yêu cầu một khóa K có độ dài bằng dữ liệu cần mã. Vấn đề chính đặt ra là quá trình tạo khóa K phù hợp trong khi vẫn đảm bảo độ an toàn của khóa.

* Giải pháp

Độ dài của dữ liệu cần mã có thể rất lớn và không cố định. Vì vậy việc tạo một khóa mã hóa có độ dài bằng dữ liệu mã là không hiệu quả. Chúng em tạo một khóa K có độ dài đủ lớn và tiến hành chia nhỏ dữ liệu để mã hóa với khóa K đó.

Trong Hình 4‑2, dữ liệu cần mã hóa D (plain text) sẽ được chia thành các khối (block) có độ dài bằng độ dài khóa K. Tiến thành xor dữ liệu với khóa theo từng byte thu được dữ liệu mã hóa (cipher text).



Hình ‑ Cơ chế mã hóa và giải mã.

Sử dụng **AesCrytoServiceProvider** (**Namespace:**  [System.Security.Cryptography](https://msdn.microsoft.com/en-us/library/system.security.cryptography(v=vs.110).aspx), **Assembly:**  System.Core (in System.Core.dll)) tạo ra khóa mã hóa (key) K từ password, IV và dữ liệu khởi tạo. Độ dài khóa K mặc định là 128 bit đảm bảo an toàn cho các dữ liệu mật theo như báo cáo của Bộ An ninh Quốc Gia Hoa Kỳ [16]. Những dữ liệu tuyệt mật (TOP SECRET) được mã hóa bằng khóa K có độ dài 256 bit.

#### Quá trình giải mã

Quá trình giải mã dữ liệu tương tự như quá trình mã hóa. Bởi bản chất của cả 2 thao tác là thực hiện một phép XOR với khóa K. Tạo khóa giải mã (tương tự khóa mã hóa) K. Trong trường hợp này khóa giải mã cũng chính là khóa mã hóa.

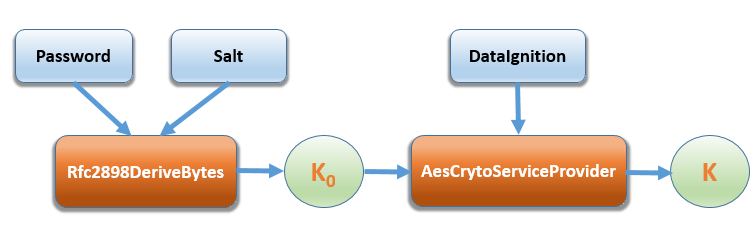
Dữ liệu được chia nhỏ dữ liệu mã hóa (cipher text) thành từng khối (block) có kích thước bằng khóa K. Sau đó, thực hiện phép xor dữ liệu mã hóa và khóa K sẽ thu được dữ liệu ban đầu D (plain text).

## Quá trình tạo khóa

Chúng em sử dụng **AesCrytoServiceProvider** mã hóa một khối dữ liệu để tạo **khóa K**. **AesCrytoServiceProvider** cần **IV** (giá trị khởi tạo ban đầu) và một thông tin bí mật gọi là **Key K0**.

Để tăng tính bảo mật, **K0** của **AesCrytoServiceProvider** sẽ được tạo thành từ 2 thành phần: **password** (mật khẩu-thông tin bí mật) và **salt** (muối-tăng tính ngẫu nhiên). Sử dụng lớp **Rfc2898DeriveBytes** với phương thức khởi tạo sử dụng 2 tham số **password** và **salt** để tạo **K0**.

Cuối cùng, **AesCrytoServiceProvider** nhận vào khóa **K0** sẽ mã hóa một khối dữ liệu (**dataIgnition** – dữ liệu khởi tạo) tạo thành khóa **K**.



Hình 4‑3 Quá trình tạo khóa mã hóa K.

Hình Hình 4‑3 trình bày quá trình tạo khóa K mã hóa và giải mã dữ liệu. Quá trình này được thực hiện bởi 2 lớp và các thông tin bí mật để đảm bảo tính bảo mật cao.

## Thông tin mã hóa (metadata)

Với mỗi file mã hóa cần chứa thông tin để giải mã. Thông tin mã hóa chứa thông tin liên quan đến thao tác mã hóa và giải mã. Vấn đề đặt ra là lưu thông tin mã hóa này chứa những gì, lưu ở đâu, như thế nào?

### Thông tin mã hóa chứa những gì?

* Vấn đề

Một file dữ liệu khi mã hóa và giải mã sẽ cần thông tin để thực hiện chính xác. Những thông tin nào là cần và đủ để đi kèm với file dữ liệu mà vẫn đảm bảo tính bảo mật.

* Giải pháp

Quá trình mã hóa và giải mã dữ liệu của chương trình đơn giản là thực hiện phép xor tuần tự khóa K và dữ liệu. Vì vậy, thông tin mã hóa sẽ phải chứa dữ liệu liên quan đến việc hình thành khóa. Vì khóa K được tạo từ quá trình mã hóa AES (Hình 4‑3) nên cần có thông tin khởi tạo ban đầu IV. Đây là dữ liệu quan trọng thứ 2 trong thông tin mã hóa.

Tiếp theo, tùy vào cách lưu trữ thông tin mã hóa mà chúng em có những thông tin thích hợp kèm theo.

### Lưu thông tin mã hóa ngay trong file được mã hóa.

Yêu cầu này đặt ra một số vấn đề quan trọng cần xử lý:

#### Ghi thêm thông tin vào file trong khi hook

* Vấn đề

Câu hỏi đặt ra là có thể ghi thêm thông tin vào file cần mã hóa hay không. Việc ghi thêm dữ liệu vào file có ảnh hưởng đến hoạt động của các ứng dụng tương tác với file dữ liệu hay không?

* Giải pháp

Hoàn toàn có thể ghi thêm dữ liệu vào một file biết trước khi chặn hàm WriteFile(). Hàm WriteFile() là hàm API chịu trách nhiệm ghi dữ liệu của hệ thống.

Cú pháp hàm WriteFile()

BOOL WINAPI WriteFile(

\_In\_ HANDLE hFile,

\_In\_ LPCVOID lpBuffer,

\_In\_ DWORD nNumberOfBytesToWrite,

\_Out\_opt\_ LPDWORD lpNumberOfBytesWritten,

\_Inout\_opt\_ LPOVERLAPPED lpOverlapped

);

Trong đó:

* hFile là handle của file hoặc I/O device được ghi xuống.
* lpBuffer là con trỏ tới dữ liệu được ghi xuống file hoặc device.
* nNumberOfBytesToWrite là số byte dữ liệu được ghi.
* lpNumberOfBytesWritten là con trỏ tới biến lưu số byte dữ liệu được ghi xuống thành công.
* lpOverlapped là con trỏ tới một cấu trúc nếu file được mở với cờ **FILE\_FLAG\_OVERLAPPED, ngoài ra tham số này có thể bằng NULL**

Khi thay đổi dữ liệu ghi xuống bởi hàm WriteFile() cần thay đổi 2 tham số lpBuffer và nNumberOfBytesToWrite. Thao tác ghi dữ liệu sẽ thực hiện bình thường. Cần lưu ý là hàm WriteFile không thay đổi được vị trí bắt đầu ghi dữ liệu (thường gọi là offset).

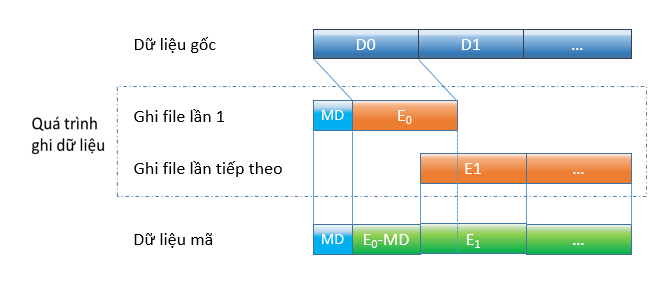
#### Lưu thông tin mã hóa ở vị trí nào của file?

Khi xác định có thể lưu thông tin mã hóa vào file, quyết định vị trí của thông tin mã hóa rất quan trọng. Bởi vì quá trình ghi dữ liệu xuống một file (thực chất là gọi hàm WriteFile()) được chia nhỏ. Khi đó, một thay đổi nhỏ trong một lần ghi xuống file cũng có thể ảnh hưởng đến cả quá trình ghi file và tạo ra file lỗi hoặc crash ứng dụng thực hiện WriteFile().

Giả sử một file F được ghi xuống thiết bị USB (hoặc ổ đĩa) có dữ liệu được chia nhỏ thành N phần liên tiếp (N>1), mỗi phần có kích thước là Di byte. Mỗi phần dữ liệu này được ghi xuống bởi hàm WriteFile(). Hàm WriteFile nhận dữ liệu, mã hóa và thực hiện ghi xuống thiết bị USB. Dữ liệu mã hóa E có kích thước đúng bằng dữ liệu gốc, vì thao tác mã hóa thực chất là thực hiện phép xor.

**Trường hợp 1:** thêm MD byte dữ liệu vào đầu file F (MD < D0). Lúc này tổng kích thước cần ghi xuống lần đầu tiên là T = E­0­ + MD byte (E0 = D0). Để lưu được T byte, hàm WriteFile() phải thay đổi tham số kích thước được ghi để có thể lưu thành công.

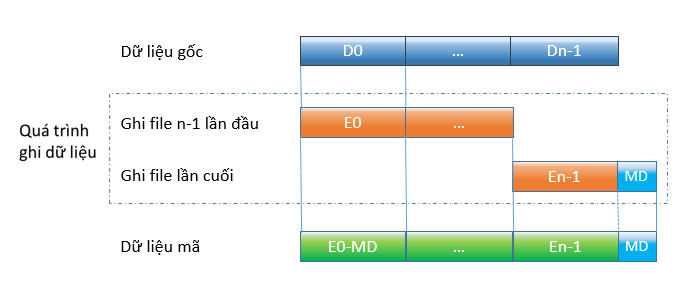
Trong quá trình can thiệp hàm WriteFile() có thể thay đổi giá trị tham số kích thước file được ghi để thực hiện thao tác này. Theo đó, vị trí bắt đầu ghi tiếp theo sẽ phải dịch một khoảng bằng D0 – MD = E0 – MD byte. Thế nhưng, hàm WriteFile không có tham số lưu vị trí bắt đầu ghi tiếp theo (offset). Lần ghi dữ liệu thứ 2 (dữ liệu D­1), hệ thống giữ nguyên vị trí ghi ban đầu dẫn tới việc dữ liệu ở lần ghi đầu tiên bị ghi đè (Hình 4‑4).



Hình ‑ Lưu thông tin mã hóa vào đầu file mã hóa.

**Trường hợp 2:** thêm MD byte dữ liệu vào cuối file F. Trong lần ghi dữ liệu cuối cùng (dữ liệu DN-1), chúng ta ghi tổng cộng DN-1 + MD byte. Thao tác này chỉ cần thay đổi tham số kích thước được ghi và không quan tâm đến vị trí ghi dữ liệu tiếp theo (vì là lần ghi cuối cùng).

Khi đọc dữ liệu file F’ (file F được mã hóa), chỉ cần đọc MD byte thông tin mã hóa ở cuối file và tiến hành giải mã dựa theo thông tin mã hóa này (Hình 4‑5).



Hình ‑ Lưu thông tin mã hóa vào cuối file mã hóa.

#### Microsoft Office không cho phép mở file đã mã hóa.

Đối với các file thuộc phiên bản Microsoft Office 2007 trở về sau, khi thêm một phần dữ liệu vào cuối file, các ứng dụng mở file lên không thành công và thông báo các file này đã bị lỗi. Cuối cùng, ứng dụng xác nhận người dùng có “tin tưởng” (trust) file này để mở lên và repair hay không. Chỉ khi người dùng chọn “tin tưởng” thì lúc đó file mới được mở bình thường.

Đối với các loại file thuộc các chương trình khác thì việc đó không gây ảnh hưởng gì.

### Lưu thông tin mã hóa ra một file riêng biệt.

Việc lưu thông tin mã hóa riêng biệt với file cần bảo vệ sẽ giải quyết vấn đề lỗi cấu trúc file. File lưu thông tin mã hóa (FT) có thể chứa thông tin rác và sắp xếp có cấu trúc nhằm bảo vệ thông tin bên trong.

#### Mã hóa và giải mã

Trong trường hợp này, quá trình mã hóa và giải mã đơn giản hơn rất nhiều so với giải pháp ghi thông tin mã hóa vào file cần bảo vệ.

Cả 2 thao tác mã và giải mã đều thực hiện đọc thông tin mã hóa từ file. Lọc những thông tin cần thiết (bỏ dữ liệu rác, ghép nối thông tin…) để thực thi.

Sử dụng thông tin mã hóa để tạo khóa (key). Tiến hành mã hóa và giải mã với khóa đã tạo.

#### Mất file chứa thông tin mã hóa

* Vấn dề:

Mất file chứa thông tin mã hóa sẽ không thể giải mã chính xác vì không có thông tin khởi tạo ban đầu IV.

* Giải pháp:

Hook hàm DeleteFile(), ngăn không cho người dùng xóa file thông tin mã hóa.

Vì file chứa thông tin mã hóa được ẩn đi và không tốn nhiều dung lượng bộ nhớ nên việc tồn tại của nó không quá ảnh hưởng đến công việc của người dùng.

## Kết luận

Nội dung Chương 5 đã trình bày các giải pháp bảo vệ dữ liệu được đề xuất. Trong đó có các cơ chế mã hóa và giải mã dữ liệu trong tập tin. Cơ chế này phải đảm bảo dữ liệu luôn ở dạng mã hóa khi được lưu xuống. Nội dung chương này còn trình bày quá trình phát sinh khóa mã hóa và giải mã giữ liệu. Ngoài ra, Chương 3 còn thảo luận các vấn đề liên quan đến thông tin mã hóa của tập tin.

# Kiến trúc hệ thống

* Nội dung **Error! Reference source not found.** trình bày hệ thống framework được nhóm phát triển để hiện thực hóa quy trình rút trích ontology từ WWW đã được trình bày ở **Error! Reference source not found.**. Các vấn đề và giải pháp khi xây dựng framework và hiện thực hóa quy trình cũng được trình bày trong chương này.

## Kiến trúc hệ thống

Giải pháp phần mềm được thiết kế tổng quan gồm 3 phần chính: Tương tác với người dùng, bảo vệ dữ liệu và can thiệp hệ thống.



Hình ‑ Kiến trúc hệ thống.

Trong Hình 5‑1, hệ thống được chia thành 3 phần riêng biệt:

User Interface tương tác với người dùng. Giao diện thân thiện giúp người dùng dễ dàng tùy chỉnh cấu hình.

Protect data bảo vệ dữ liệu trong quá trình ghi xuống ổ cứng hoặc các thiết bị lưu trữ khác.

Hook Module có khả năng can thiệp hệ thống, chặn và kiểm soát dữ liệu của các hàm API.

Kiến trúc từng phân hệ, các vấn đề và giải pháp liên quan được chúng em trình bày ở các phần sau.

## Phân hệ can thiệp hệ thống – Hook Module



Hình ‑ Phân hệ can thiệp hệ thống.

Hook Module có khả năng giám sát và can thiệp vào các hàm API của hệ thống để kiểm soát và xử lý dữ liệu.

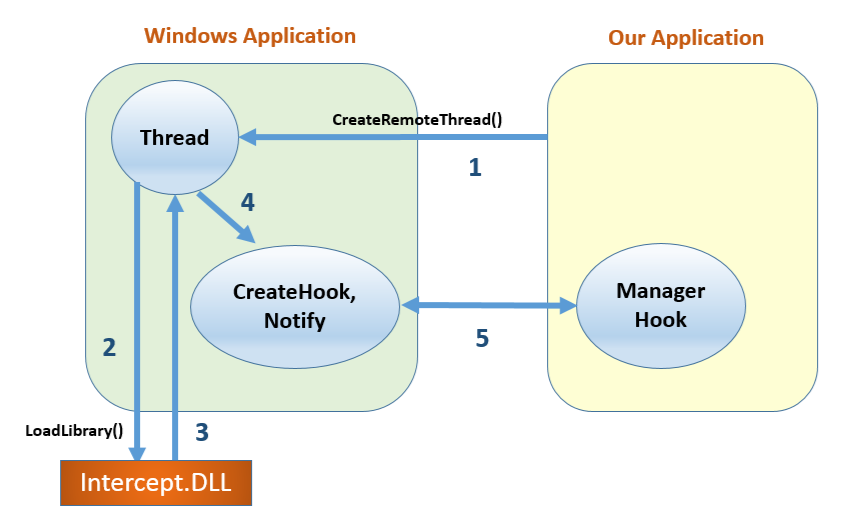
Hình 5‑2 mô tả kiến trúc Hook Module gồm 3 thành phần chính là DLL Injection, API Interception và EasyHook.

* DLL Injection Module có chức năng chèn mã DLL vào một tiến trình xác định. Từ đó thực hiệc các thao tác tiếp theo trong quá trình Hook.
* API Intercepting Module sẽ chặn hàm API và kiểm soát dữ liệu trước khi được trả về cho hệ thống hoặc các ứng dụng Windows.
* EasyHook cung cấp một số hàm hỗ trợ cho 2 module DLL Injection và API Intercepting hoạt động.

Ở ứng dụng hiện tại, chúng em sử dụng thư viện EasyHook trên nền .NET framework để hỗ trợ thực hiện công việc can thiệp hệ thống. Một số chức năng của EasyHook đã được chúng em cải thiện để đảm bảo chương trình chạy ổn định.

### DLL Injection Module

Chúng em chọn giải pháp inject DLL bằng phương pháp CreateRemoteThread. (EasyHook có hỗ trợ phương pháp này). Chương trình của chúng em sẽ tạo một tiểu trình trong các tiến trình mục tiêu và tải các DLL cần thiết vào trong tiến trình.



Hình ‑ Hoạt động can thiệp hệ thống.

Trong Hình 5‑3, ứng dụng Our Application gọi hàm CreateRemoteThread() để tạo một Thread bên trong ứng dụng Windows mục tiêu (bước 1). Thread mới được tạo sẽ gọi hàm LoadLibrary() (bước 2) và load Intercept.DLL vào ứng dụng mục tiêu (bước 3).

* Vấn đề

Tại sao lại sử dụng một phương pháp Inject DLL cục bộ (local hook). Thực hiện trên toàn cục (global hook) có khác biệt gì không? Sử dụng phương pháp nào là tốt nhất cho hệ thống?

* Giải pháp

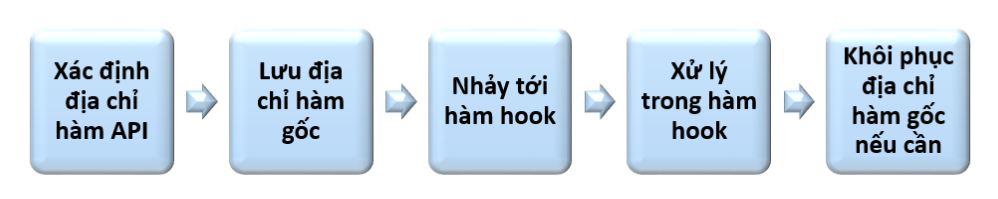
Hai giải pháp Inject DLL cục bộ và toàn cục đều mang lại kết quả như mong muốn. Đó là đưa được mã của mình vào một tiến trình xác định. Nhưng khi thực thi trên toàn cục sẽ đem lại những thao tác không cần thiết, thậm chí xảy ra lỗi.

Bản chất của Global hook là tác động lên tất cả (hoặc gần như tất cả) các tiến trình đang chạy. Những tiến trình này phần lớn sử dụng các hàm API của hệ thống và trong đó có chứa một vài hàm API mà chương trình muốn tác động. Chính vì thế, chương trình có thể tác động đến những tiến trình không liên quan đến mục đích chính và tác động đến quá trình gọi hàm của tiến trình đó. Điều này dẫn đến việc vô tình khiến hệ thống phải xử lý nhiều hơn, tăng các thao tác phải thực hiện cũng như nguy cơ xảy ra lỗi trong quá trình hook. Chưa kể đến việc unhook ở một số tiến trình xác định khi không cần thiết. Global hook chỉ có thể unhook ở tất cả các tiến trình.

Trong khi đó, Local hook giải quyết được vấn đề chính gặp phải của Global hook về việc tác động lên tiến trình cần thiết và linh hoạt trong các thao tác hook.

### API Intercepting Module

Trong Hình 5‑3, sau khi Thread mới được tạo ở một Windows Application, Thread này sẽ bắt đầu can thiệp hệ thống (bước 4) và tiến hành trao đổi dữ liệu với Our Application (bước 5).



Hình ‑ Quá trình can thiệp hàm API.

Chương trình sử dụng cơ chế intercept API phổ biến là thay thế code của hàm API.

Hình 5‑4 trình bày quá trình can thiệp hệ thống của Our Application. Bước đầu tiên, chương trình xác định địa chỉ hàm API gốc lưu trong bộ nhớ. Tiếp theo, địa chỉ hàm gốc sẽ được lưu lại để có thể gọi thực thi trong quá trình can thiệp và khôi phục lại ở bước cuối cùng. Sau khi đảm bảo địa chỉ hàm gốc được lưu giữ lại, chương trình ghi một lệnh nhảy JMP tới địa chỉ hàm hook. Hàm hook này có signature giống với hàm API ban đầu. Lúc này, chương trình có thể lấy được dữ liệu được truyền vào hàm API và xử lý chúng. Bước cuối cùng trong quá trình này là khôi phục lại địa chỉ hàm API gốc để đảm bảo ứng dụng mục tiêu hoạt động bình thường.

### Hoạt động của ứng dụng.

#### Giải pháp tiếp cận

* Vấn đề

Cần xác định những hàm API liên quan đến các thao tác lưu trữ dữ liệu. Sau đó tiến hành can thiệp trong quá trình thực thi của các hàm API này để bảo vệ dữ liệu.

* Giải pháp

Quá trình đọc và ghi file của các ứng dụng trên hệ điều hành Windows 7 hiện tại đều thông qua 2 hàm API chính. Đó là ReadFile và WriteFile. Chúng em đã can thiệp 2 API này để xử lý việc mã hóa và giải mã file.

**Quá trình xử lý WriteFile().**

BOOL WINAPI WriteFile(

\_In\_ HANDLE hFile,

\_In\_ LPCVOID lpBuffer,

\_In\_ DWORD nNumberOfBytesToWrite,

\_Out\_opt\_ LPDWORD lpNumberOfBytesWritten,

\_Inout\_opt\_ LPOVERLAPPED lpOverlapped

);

Các ứng dụng lưu dữ liệu xuống máy tính và thiết bị lưu trữ sẽ sử dụng hàm WriteFile() của hệ thống. Khi can thiệp hàm WriteFile chúng em lấy được các thông tin cơ bản về tên file (đầy đủ đường dẫn từ handle file), con trỏ buffer chứa dữ liệu lưu xuống file, số byte được ghi, …

Khi có được tên file, chương trình sẽ kiểm tra file có được hỗ trợ mã hóa hay không? Sau đó, tiến hành đọc thông tin mã hóa từ file metadata và tạo khóa K mã hóa. Nếu không đọc được thông tin mã hóa, chương trình tự động pháp sinh thông tin mã hóa và khóa K tương ứng. Thông tin mã hóa này sẽ được lưu xuống file phục vụ cho quá trình giải mã.

Thao tác cuối cùng xử lý trong hàm WriteFile() là mã hóa dữ liệu trong buffer. Buffer được chia thành nhiều Block có độ dài bằng khóa K và được mã hóa tuần tự cho đến Block cuối cùng.

**Quá trình xử lý ReadFile().**

BOOL WINAPI ReadFile(

\_In\_ HANDLE hFile,

\_Out\_ LPVOID lpBuffer,

\_In\_ DWORD nNumberOfBytesToRead,

\_Out\_opt\_ LPDWORD lpNumberOfBytesRead,

\_Inout\_opt\_ LPOVERLAPPED lpOverlapped

);

Các ứng dụng khi mớ file sẽ nhận được dữ liệu trả về từ hàm ReadFile(). Chúng em tiến hành can thiệp và xử lý hàm ReadFile() để giải mã dữ liệu. Trong hàm ReadFile() chứa các thông tin cơ bản phục vụ cho thao tác đọc dữ liệu từ file đó là tên file, con trỏ buffer dữ liệu được đọc lên, số byte dữ liệu đọc,…

Con trỏ buffer trong hàm ReadFile() chưa có dữ liệu, đây là điểm khác biệt lớn so với hàm WriteFile(). Vì vậy, chương trình cần gọi hàm gốc ReadFile() để lấy dữ liệu được đọc lên và giải mã trước khi hàm trả về.

Tiếp theo, chương trình kiểm tra phần mở rộng của file để xác định có tiến hành giải mã hay không. Thao tác cuối cùng tương tự như quá trình xử lý WriteFile, chương trình tìm thông tin mã hóa tương ứng với file trong bộ nhớ hoặc từ file chứa thông tin mã hóa. Cuối cùng, dữ liệu trong vùng nhớ con trỏ buffer trỏ tới đã chứa dữ liệu được giải mã và được trả về cho ứng dụng đọc file.

#### Mở một file đã được mã hóa

* Vấn đề

Khi một file được ứng dụng mã hóa, tất cả dữ liệu trong file đều được mã theo cơ chế được quy định riêng trong ứng dụng. Những ứng dụng khác sẽ không thể đọc được đúng dữ liệu trong file. Lúc này cần xử lý hàm ReadFile() để trả về dữ liệu ban đầu có thể đọc hiểu cho các ứng dụng đọc file. Vấn đề đặt ra là không thể xử lý hàm ReadFile() khi đọc tất cả các file một cách giống nhau. Bởi vì có thể xuất hiện những file chưa được mã hóa, chương trình sẽ đọc sai dữ liệu của những file này.

Khi người dùng mở một file hay một ứng dụng để xử lý các loại file cần bảo vệ được quy định thì chúng ta cần can thiệp vào các ứng dụng tương ứng để xử lý dữ liệu. Làm sao để biết được khi nào ứng dụng được chạy để xử lý các file tương ứng?

* Giải pháp

Hiện tại thì trên Windows 7, ứng dụng Windows Explorer được sử dụng để quản lý file và các ứng dụng đọc ghi file thông qua hàm API CreateProcessW(). Vì thế, ứng dụng cần phải can thiệp vào hàm API này để tiến trình explorer.exe kiểm soát và chèn DLL cần thiết vào các tiến trình/ứng dụng thao tác với những file cần được bảo vệ.

**Quá trình xử lý CreateProcess()**

BOOL WINAPI CreateProcess(

\_In\_opt\_ LPCTSTR lpApplicationName,

\_Inout\_opt\_ LPTSTR lpCommandLine,

\_In\_opt\_ LPSECURITY\_ATTRIBUTES lpProcessAttributes,

\_In\_opt\_ LPSECURITY\_ATTRIBUTES lpThreadAttributes,

\_In\_ BOOL bInheritHandles,

\_In\_ DWORD dwCreationFlags,

\_In\_opt\_ LPVOID lpEnvironment,

\_In\_opt\_ LPCTSTR lpCurrentDirectory,

\_In\_ LPSTARTUPINFO lpStartupInfo,

\_Out\_ LPPROCESS\_INFORMATION lpProcessInformation

);

Hàm CreateProcess() khi được gọi sẽ cần có các tham số: tên ứng dụng muốn khởi chạy (gọi là ứng dụng mục tiêu), lệnh command line, các tham số về các tiến trình và tiểu trình, cờ thông báo trạng thái ứng dụng khi được tạo… Khi can thiệp hàm CreateProcess(), chúng em sẽ lấy được các tham số cơ bản này.

Điều quan trọng trong quá trình xử lý hàm CreateProcess() là phải lấy được ProcessId của ứng dụng muốn tạo và tiến hành quá trình chèn Inject.DLL vào ứng dụng đó. Sau đó, Inject.DLL này sẽ tiến hành can thiệp vào tất cả các thao tác đọc ghi file của ứng dụng mới được tạo và xử lý dữ liệu.

**Trạng thái của ứng dụng được tạo**

* Vấn đề

Nhưng một vấn đề nghiêm trọng gặp phải khi khởi chạy một ứng dụng đó là quá trình đọc file và chèn DLL tiến hành song song với nhau. Nhiều trường hợp, quá trình đọc file nhanh hơn dẫn đến việc DLL chưa kịp chèn vào ứng dụng mục tiêu để xử lý dữ liệu được đọc lên. Điều này khiến quá trình mở file bị lỗi.

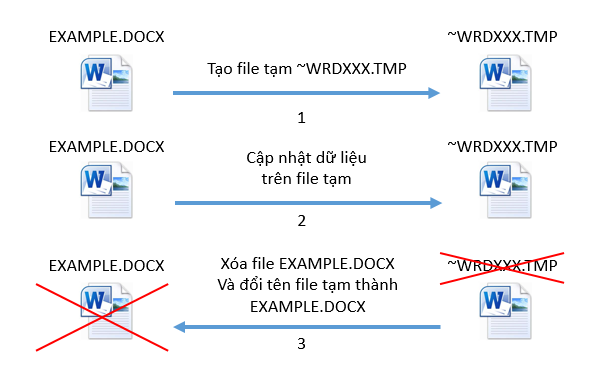
* Giải pháp

Khi can thiệp hàm CreateProcess(), chuyển trạng thái ứng dụng khi khởi chạy về CREATE\_SUSPENDED. Tiểu trình chính trong tiến trình/ứng dụng mục tiêu sẽ bị đình chỉ. Sau khi chèn DLL vào ứng dụng, tiểu trình chính sẽ được tiếp tục chạy. Giải pháp này đảm bảo ứng dụng mục tiêu được chèn DLL trước khi thực hiện các thao tác liên quan đến dữ liệu trong file cần bảo vệ.

#### Một số thao tác xử lý ngoại lệ

* Vấn đề

Một số ứng dụng Office như Microsoft Word, Excel … không sử dụng hàm WriteFile() khi thay đổi dữ liệu một file đã tồn tại (thao tác cập nhật nội dung) mà lại có cách xử lý khác được minh họa bằng hình vẽ sau:



Hình ‑ Quá trình cập nhật dữ liệu của file .docx

Đầu tiên, mỗi khi có sự thay đổi trên file, ứng dụng office sẽ tạo một file tạm có phần mở rộng là .tmp (bước 1). Với mỗi thao tác cập nhật dữ liệu trên file EXAMPLE.DOCX, những dữ liệu đó cũng sẽ được cập nhật lên file tạm (bước 2). Khi kết thúc quá trình làm việc, ứng dụng sẽ xóa file EXAMPLE.DOCX và đổi tên file tạm thành tên file ban đầu – EXAMPLE.DOCX (bước 3).

Vì trong thao tác này, ứng dụng Office sử dụng hàm WriteFile() để ghi dữ liệu xuống file tạm nên suy nghĩ ban đầu của chúng em xác định xử lý trên hàm WriteFile() nhưng bình thường. Nhưng vấn đề gặp phải là vùng nhớ để ghi file tạm không thể truy xuất được. Chúng em đã thử thay đổi quyền truy cập vùng nhớ bằng hàm VirtualProtect() nhưng không thành công. Có cách nào thay đổi được dữ liệu trên file cuối cùng được lưu lại?

* Giải pháp

Trong quá trình theo dõi thao tác cập nhật dữ liệu trên file, chúng em phát hiện ứng dụng Office sử dụng hàm ReplaceFile() để tiến hành đổi tên file.

BOOL WINAPI ReplaceFile(

\_In\_ LPCTSTR lpReplacedFileName,

\_In\_ LPCTSTR lpReplacementFileName,

\_In\_opt\_ LPCTSTR lpBackupFileName,

\_In\_ DWORD dwReplaceFlags,

\_Reserved\_ LPVOID lpExclude,

\_Reserved\_ LPVOID lpReserved

);

Khi file được cập nhật dữ liệu nằm trên thiết bị lưu trữ ngoài (USB, thẻ nhớ..), hàm MoveFileExW() được gọi để thực hiện công việc tương tự như hàm ReplaceFile() trên ổ cứng.

BOOL WINAPI MoveFileEx(

\_In\_ LPCTSTR lpExistingFileName,

\_In\_opt\_ LPCTSTR lpNewFileName,

\_In\_ DWORD dwFlags

);

**Quá trình xử lý ReplaceFile, MoveFileEx.**

Mục đích của 2 hàm ReplaceFile() và MoveFileEx() cũng giống như hàm WriteFile() là ghi dữ liệu xuống file. Nhưng điểm khác biệt lớn nhất ở đây là số lần thực hiện của 2 hàm này: chỉ được gọi 1 lần duy nhất để thay thế file ban đầu bằng file tạm. Vì thế, tất cả dữ liệu sẽ được mã hóa 1 lần duy nhất trong khi can thiệp 2 hàm này. Chính vì lý do này, quá trình ghi file xuống có thể chậm đối với những file có dung lượng lớn. Nhưng với sự phát triển của bộ vi xử lý thì thời gian làm chậm này cũng không đáng kể và có thể giảm đi theo thời gian phát triển công nghệ sản suất CPU.

#### Bảo vệ file thông tin mã hóa – file metadata

* Vấn đề

Việc giải mã dữ liệu khi đọc file phụ thuộc vào thông tin mã hóa. Thông tin này có thể lưu trữ trực tiếp trên bộ nhớ chương trình hoặc đọc từ file metadata. Khi file lưu thông tin mã hóa bị mất và hư hại.

* Giải pháp

Để ngăn chặn người dùng xóa các file data cần thiết cho hệ thống, giải pháp hiện tại cần phải can thiệp hàm API DeleteFile().

**Quá trình xử lý DeleteFile**

BOOL WINAPI DeleteFile(

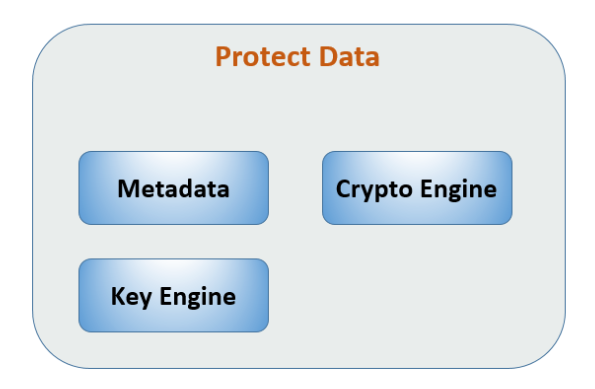
\_In\_ LPCTSTR lpFileName

);

Khi can thiệp vào hàm DeleteFile(), chương trình kiểm tra tên file (đầy đủ đường dẫn) có phải là file chưa thông tin mã hóa hay không? Nếu hàm DeleteFile() nhận tham số là tên file thông tin mã hóa, chương trình sẽ trả về kết quả thực hiện không thành công. Khi đó, file chứa thông tin mã hóa sẽ không bị xóa bỏ.

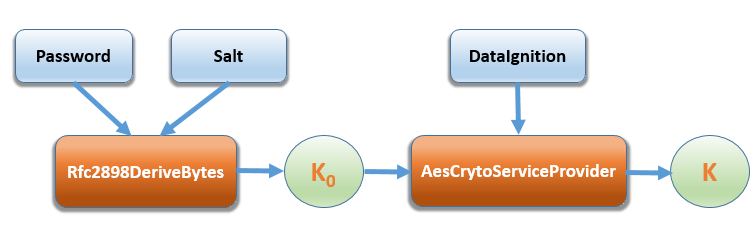
Để chống sửa dữ liệu trong file thông tin mã hóa, chúng em xử lý trong hàm WriteFile(). Với thao tác kiểm tra tên file giống như với hàm DeleteFile(), chương trình ngăn chặn việc ghi file từ bên ngoài đối với file chứa thông tin mã hóa.

## Phân hệ bảo vệ dữ liệu – Protect data



Hình ‑ Phân hệ bảo vệ dữ liệu.

### Tạo Key mã hóa – Key Engine



Hình ‑ Quá trình tạo khóa mã hóa K.

Quy trình tạo khóa K giải mã được tiến hành qua 2 bước nhằm tăng tính bảo mật.

Sử dụng lớp Rfc2898DeriveBytes để tạo khóa K0. Khóa K0 này có độ dài bất kỳ và có thể không có giới hạn. Đặc điểm này thuận lợi cho việc tạo khóa cho nhiều thuật toán mã hóa.

Sử dụng lớp AesCrytoServiceProvider nhận khóa K0 làm khóa mã hóa. Sau đó, thể hiện của lớp này tiến hành mã hóa một dữ liệu khởi tạo bí mật để tạo thành khóa K.

### Thông tin mã hóa – metadata

Chưa các thông tin cần thiết phục vụ cho việc mã hóa, giải mã dữ liệu. Ngoài ra còn có các thông tin liên quan đến quyền truy cập dữ liệu.

Những thông tin cần thiết bao gồm : tên file mã hóa, IV tạo khóa và bản hash dữ liệu bí mật (Password người dùng hoặc tên domain).

Tên file mã hóa: dùng để xác định file đã được mã hóa hay chưa. Nếu file chưa mã hóa thì không tiến hành xử lý tiếp mà trả gọi lại hàm API ban đầu. Ngược lại, chương trình tiến hành giải mã file và trả dữ liệu về cho các ứng dụng đọc file.

IV: thông tin khởi tạo khóa K mã hóa và giải mã. IV được tạo riêng cho từng file, không trùng lặp và không cần thiết phải bảo mật.

Bản hash dữ liệu bí mật: dùng để chứng thực người dùng. Người dùng có chính xác password (hoặc tên domain được lấy tự động trên máy tính) sẽ được chương trình chứng thực bằng cách so sánh với bản hash lưu trong file thông tin mã hóa. Ngay cả khi thay thế bản hash này, kẻ xấu cũng không thể giải mã được dữ liệu. Vì chính dữ liệu bí mật dùng để hash này được dùng để tạo khóa K giải mã dữ liệu. Chỉ khi nhận đúng dữ liệu bí mật, chương trình sẽ tạo đúng khóa K và giải mã chính xác dữ liệu trong file mã hóa.

### Mã hóa và giải mã – Crypto Engine

Crypto Engine chịu trách nhiệm mã hóa và giải mã dữ liệu. Thao tác mã hóa và giải mã chỉ đơn giản là thực hiện phép XOR của dữ liệu với khóa K. Do thao tác tạo khóa sử dụng thuật toán mã hóa đối xứng và thao tác mã hóa đơn giản giảm thời gian xử lý và tăng hiệu suất của chương trình.

* Vấn đề

Thao tác mã hóa và giải mã thực hiện phép XOR. Vậy có khả năng kẻ xấu thực hiện phép XOR giữ dữ liệu ban đầu và dữ liệu mã hóa để tìm ra khóa K. Vậy cơ chế này có an toàn ?

* Giải pháp

Cơ chế mã hóa này vẫn đảm bảo an toàn. Bởi mỗi file dữ liệu bảo vệ có một khóa K riêng biệt nhờ vào IV được sinh ngẫu nhiên. Với thuật toán mã hóa AES được sử dụng trong quá trình tạo khóa K có độ dài khóa là 128 bit đảm bảo an toàn theo như báo cáo của Cơ quan an ninh Quốc gia Hoa Kỳ [16]. Vì thế, ngay cả khi biết được khóa K thì kẻ xấu cũng khó có thể giải mã được thông tin bí mật của người dùng (password).

* Vấn đề

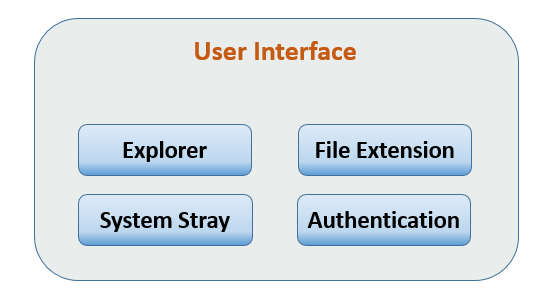
Một tập tin mã hóa muốn truy cập để có thể đọc được dữ liệu ban đầu (plain text) bắt buộc phải giải mã. Khi tập tin được giải mã có thể làm mất tính bảo mật của ứng dụng vì dữ liệu tồn tại ở dạng chưa mã hóa. Biện pháp nào để đọc dữ liệu mà không cần giải mã ra tập tin khác?

* Giải pháp

Thao tác mã hóa và giải mã thực hiện ngay trong bộ nhớ. Hệ thống sử dụng các hàm ReadFile() và WriteFile() để đọc và ghi dữ liệu đối với tập tin. Trong quá trình ghi dữ liệu, chương trình sẽ thực hiện các thao tác mã hóa trong hàm WriteFile() để đảm bảo dữ liệu được bảo vệ khi lưu trữ xuống tập tin. Khi một ứng dụng muốn đọc dữ liệu từ một tập tin mã hóa, chương trình sẽ can thiệp hàm ReadFile() trong ứng dụng đọc tập tin và giải mã dữ liệu ngay trong bộ nhớ để trả về cho ứng dụng này dữ liệu ban đầu khi chưa mã hóa (plain text).

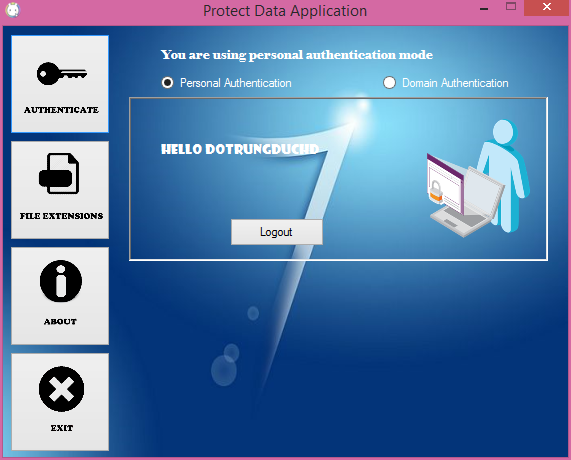
Một số ứng dụng đặc biệt còn sử dụng các hàm API khác để hỗ trợ đọc và ghi dữ liệu xuống tập tin. Ví dụ như Microsoft Word sử dụng hàm ReplaceFile() khi cập nhật thông tin trong tập tin hoặc thay thế một tập tin đã có sẵn. Trường hợp này, chúng em cũng can thiệp và xử lý dữ liệu trong quá trình những hàm API này được thực thi.

## Phân hệ tương tác với người dùng – User Interface



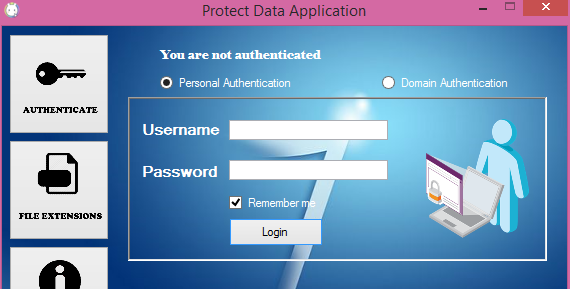
Hình ‑ Phân hệ tương tác với người dùng.

Cung cấp giao diện thân thiện, đơn giản với người dùng.



Hình ‑ Giao diện chính của ứng dụng.

### Chứng thực người dùng – Authentication

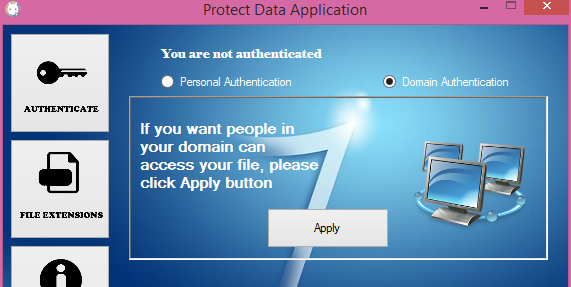


Hình ‑ Giao diện chứng thực người dùng các nhân.

Cho phép người dùng lựa chọn kiểu chứng thực.

Đối với người dùng sử dụng cho mục đích cá nhân, họ có thể dùng ID và Password để xác nhận bảo vệ. Password của người dùng sẽ được sử dụng như một thành phần tạo khóa K mã hóa và giải mã dữ liệu. Như vậy, chỉ người nào có Password mới có thể giải mã được dữ liệu trong file đã mã hóa.

Chứng thực trong một Actity Domain với các chức năng chia sẻ dữ liệu trong cùng một domain. Thông tin dùng để kiểm tra là tên domain. Chương trình sẽ tự động lấy dữ liệu về domain trên máy tính và kiểm tra với thông tin được ghi trên file thông tin mã hóa. Vì thông tin domain được chương trình lấy tự động nên ngăn chặn được trường hợp giả mạo thông tin domain. Ngay cả khi thông tin domain bị thay đổi, chương trình cũng ngăn chặn sự thay đổi trên file thông tin mã hóa làm giảm thiểu khả năng khai thác dữ liệu trái phép.



Hình ‑ Giao diện chứng thực người dùng trong Active Domain.

### Các loại file hỗ trợ bảo vệ - File Extension

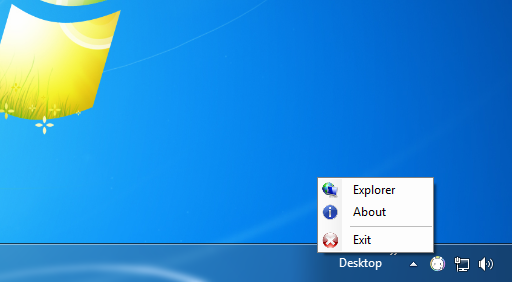
Chương trình hỗ trợ người dùng chọn các loại file muốn bảo vệ. Hiện tại, danh sách gồm những định dạng phổ biến xuất hiện trong văn phòng như doc, xls, ppt, pdf, rar, zip và những định dạng mới nhất của Microsoft Office là docx, xlsx, pptx.



Hình ‑ Giao diện tùy chỉnh loại tập tin được hỗ trợ.

### Ứng dụng chạy nền – System Stray

Ứng dụng có khả năng khởi động cùng Windows. Sau khi thu nhỏ sẽ chuyển về dạng hiển thị dưới System Stray trả lại không gian trên Taskbar cho các ứng dụng khác.



Hình ‑ Giao diện thu nhỏ của ứng dụng.

# Kết luận

* Nội dung của Chương 7 trình bày các kết quả đạt được và hướng phát triển của đề tài.

## Các kết quả đạt được

Báo cáo đã tìm hiểu thực trạng thất thoát dữ liệu hiện nay. Các kết quả phân tích cho thấy ngày càng có nhiều dữ liệu mật bị rò rì. Trong đó, tỷ lệ dữ liệu thất thoát trên máy tính và các thiết bị lưu trữ là không hề nhỏ. Những số liệu ở Chương 1 đã đặt ra nhu cầu cần có những biện pháp bảo vệ dữ liệu hiệu quả trong bối cảnh bùng nổ công nghệ thông tin.

Qua khảo sát các phần mềm bảo vệ dữ liệu phổ biến, nhóm đã chỉ ra các đặc điểm cần khắc phục về mặt tương tác với người dùng, giá cả, khó khăn khi cài đặt và triển khai… Với những phân tích trên các mặt còn hạn chế của các phần mềm đi trước, ý tưởng tạo một sản phẩm thân thiện và tạo cảm giác thoải mái cho người dùng đã được hiện thực hóa thành phần mềm.

Sau khi phân tích các cách can thiệp hệ thống, chúng em đã chọn ra một giải pháp thích hợp nhất sử dụng trong sản phẩm phần mềm. Giải pháp này đảm bảo hiệu suất xử lý và giảm thiểu các thao tác dư thừa khi can thiệp hệ thống.

Trong quá trình tìm hiểu, chúng em đề xuất một giải pháp bảo vệ dữ liệu đơn giản nhưng vẫn đảm bảo tính bảo mật. Hơn thế nữa, thao tác mã hóa và giải mã đơn giản giúp giảm thời gian xử lý, tạo cho người dùng cảm giác như đang thao tác trên máy tính bình thường. Đây cũng cũng là một trong những mục đích chính của chúng em khi thực hiện đề tài.

Cuối cùng, từ những nghiên cứu đạt được, chúng em đã hiện thực hóa giải pháp thành một sản phẩm phần mềm tương đối hoàn thiện. Phần mềm cung cấp đầy đủ chứng năng chính mã hóa và giải mã dữ liệu. Ngoài ra, phần mềm còn cho phép người dùng tùy chỉnh một số cấu hình đơn giản như đăng nhập, chọn chế độ mã hóa, chọn loại file cần bảo vệ. Với giao diện đơn giản, người dùng hoàn toàn có thể sử dụng thành thạo các thao tác trên giao diện. Chương trình có thể khởi động cùng Windows và chạy ẩn dưới System Stray.

## Hướng phát triển của đề tài

Với những kết quả đạt được, đề tài có các hướng phát triển:

* Hoàn thiện và mở rộng các chức năng hiện có của ứng dụng.

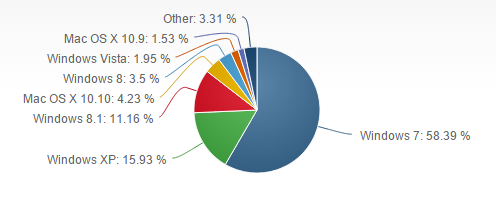
Danh mục tài liệu tham khảo

|  |  |
| --- | --- |
| [1] | InfoWatch, "Global Data Leakages & Insider Threats Report - 2012," InfoWatch Research Center, 2012. |
| [2] | InfoWatch, "Global Data Leakage Report - 2013," InfoWatch Analytical Center, 2014. |
| [3] | InfoWatch, "Infowatch's Global Data Leakage Report for First Half of 2014," 2014. |
| [4] | D. C. Timon, L. Feng and K. Irwin, "Managing knowledge on the Web - Extracting ontology from HTML Web," *Decision Support Systems,* vol. 47, no. 4, pp. 319-331, November 2009. |
| [5] | McAfee, "McAfee for Small Business," [Online]. Available: http://www.shopmcafee.com/store/mfesmb/en\_US/DisplayHomePage?CID=MFE-MHP102. |
| [6] | McAfee, "McAfee for Business," [Online]. Available: http://www.mcafee.com/us/business-home.aspx?CID=MFEen-usMHP002. |
| [7] | TrendMicro, "Worry-Free Business Security," [Online]. Available: http://www.trendmicro.com/us/small-business/product-security/index.html. |
| [8] | TrendMicro, "Enterprise Security Software," [Online]. Available: Worry-Free Business Security. |
| [9] | TrendMicro, "Integrated Data Loss Prevention," [Online]. Available: http://www.trendmicro.com/us/enterprise/data-protection/data-loss-prevention/. |
| [10] | Symantec, "Products & Solutions," [Online]. Available: http://www.symantec.com/products-solutions/. |
| [11] | Symantec, "Data Loss Prevention," [Online]. Available: http://www.symantec.com/data-leak-prevention/. |
| [12] | EMC, "RSA DATA LOSS PREVENTION," [Online]. Available: http://www.emc.com/security/rsa-data-loss-prevention.htm. |
| [13] | WebSense, "TRITON APX Core Products," [Online]. Available: http://www.websense.com/content/triton-apx.aspx?intcmp=nav-mm-products-core-apx. |
| [14] | I. Ivanov, "API hooking revealed," 2 12 2002. [Online]. Available: http://www.codeproject.com/Articles/2082/API-hooking-revealed. |
| [15] | Wikipedia, "http://en.wikipedia.org," [Online]. Available: http://en.wikipedia.org/wiki/Portable\_Executable. |
| [16] | NSA, "https://www.nsa.gov/," 25 9 2014. [Online]. Available: https://www.nsa.gov/ia/programs/suiteb\_cryptography/index.shtml. |
| [17] | J. Xing and T. Ah-Hwee, "Mining Ontological Knowledge from Domain-Specific Text Documents," in *ICDM Proceedings of the Fifth IEEE International Conference on Data Mining*, Washington, DC, 2005. |
| [18] | S. York, B. Stephan, H. Peter, H. Jens and O. Daniel, "The SWRC Ontology - Semantic Web for Research Communities," in *Progress in Artificial Intelligence, 12th Portuguese Conference on Artificial Intelligence*, 2005. |
| [19] | A. B. Williams and C. Tsatsoulis, "An Instance-based Approach for Identifying Candidate Ontology Relations within a Multi-Agent System," in *Proceedings of the First Workshop on Ontology Learning OL'2000*, Berlin, 2000. |
| [20] | Wikipedia, "https://en.wikipedia.org/," [Online]. Available: https://en.wikipedia.org/wiki/Advanced\_Encryption\_Standard#Known\_attacks. |

PHỤ LỤC

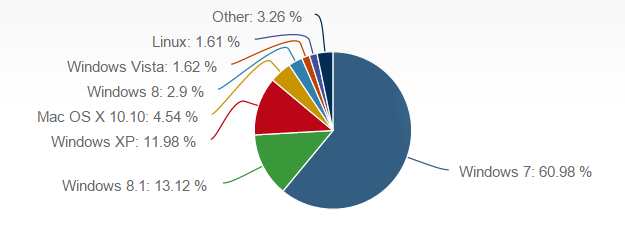
1. Hệ điều hành : Windows 7.

Net Applications hoạt động từ 1999 và trở thành website phân tích tin cậy. Theo số liệu phân tích tháng 4 năm 2015



Hình phụ lục 1. Tỷ lệ khảo sát thị phần hệ điều hành tháng 4-2015

Biểu đồ cho thấy số lượng người dùng windows 7 hiện nay rất lớn (58.39%). Windows 7 có tỷ lệ vượt trội so với 2 hệ điều hành xếp ngay sau là Windows XP (15.93%) và Windows 8.1 (11.16%).



Hình phụ lục 2. Tỷ lệ khảo sát thị phần hệ điều hành tháng 7-2015

Biểu đồ trên cho thấy thị phần Windows 7 vẫn đang có dấu hiệu tăng. Tỷ lệ của windows 7 tăng từ 58.39% lên 60.98% trong vòng 3 tháng gần đây.

Chính vì phần lớn người dùng vẫn ưa thích sử dụng Windows 7. Nhóm nghiên cứu đã chọn Windows 7 làm đối tượng nghiên cứu và thực nghiệm.