

HỌC VIỆN KỸ THUẬT MẬT MÃ  
AN TOÀN THÔNG TIN  
-----



**BÀI TẬP LỚN**

**Môn:**

**Phòng chống và điều tra tội phạm mạng máy tính**

**ĐỀ TÀI: Window Memory Analysis**

Khoa: An toàn thông tin

Chuyên ngành: An toàn thông tin

*Sinh viên thực hiện:*

**Trần Diệu Linh**

**Trương Văn Phong**

**Hoàng Huy Ngọc**

**Phạm Thị Bích**

Lớp: AT12L02

*Người hướng dẫn:*

**Giáo viên: Lại Minh Tuấn**

Khoa An toàn thông tin – Học viện Kỹ thuật mật mã

Hà Nội, 2019

## MỤC LỤC

<b>Chapter 3: Windows Forensic Analysis.....</b>	<b>6</b>
<b>1. Windows Memory Analysis .....</b>	<b>6</b>
1.1. Giới thiệu .....	6
1.2. Lược sử.....	7
1.3. Collecting Process Memory .....	10
1.4. Dumping Physical Memory .....	13
1.4.1. Nigilant32 .....	14
1.4.2. ProDiscover.....	15
1.4.3. KnTDD .....	15
1.4.4. MDD.....	18
1.4.5. Win32dd.....	19
1.4.6. Memoryze .....	21
1.4.7. Winen.....	21
1.4.8. Fastdump .....	23
1.4.9. F-Response .....	24
1.4.10. Alternative Approaches for Dumping Physical Memory.....	33
1.4.11. Hardware Devices .....	33
1.4.12. FireWire .....	34
1.4.13. Crash Dumps .....	36
1.4.14. Virtualization .....	40
1.4.15. Hibernation File .....	42
1.4.16. Analyzing a Physical Memory Dump .....	43
1.4.17. Determining .....	44
1.4.18. Process Basics .....	48
1.4.19. EProcess Structure .....	48
1.4.20. Process Creation Mechanism.....	51
1.4.21. Parsing Memory Dump Contents .....	52
1.4.22. Lspd.pl.....	57
1.4.23. Volatility Framework .....	61
1.4.24. Memoryze .....	68
1.4.25. HBGary Responder.....	72
1.4.26. Parsing Process Memory .....	77
1.4.27. Extracting the Process Image .....	80
1.4.28. Memory Dump Analysis and the Page File .....	87
1.4.29. Pool Allocations.....	87
1.4.30. Solutions Fast Track .....	89

## **Danh mục hình ảnh**

Hình 1: Một phần GUI Nigilant32 .....	14
Hình 2: Trích đoạn Hộp thoại Chụp từ ProDiscover IR.....	15
Hình 3: Ổ đĩa hệ thống Windows từ xa được kết nối với F: \ .....	26
Hình 4: Giao diện người dùng FEMC .....	27
Hình 5: Đăng nhập vào F-Feedback EE thông qua Giao diện người dùng FEMC ..	28
Hình 6: Cấu hình F-Feedback EE thông qua Giao diện người dùng FEMC.....	28
Hình 7: F-Response EE Connection to Remote RAM Seen via Disk Management MMC .....	29
Hình 8: Chọn RAM hệ thống từ xa thông qua FTK Imager.....	30
Hình 9: RAM được chọn trong cây bằng chứng FTK Imager.....	30
Hình 10: Các mục menu để tạm dừng một phiên trong VMware Workstation 6.5.	41
Hình 11: GUI Audit Viewer hiển thị thay đổi cấu hình .....	70
Hình 12: Process tree trong GUI Audit Viewer .....	70
Hình 13: Tab hiển thị chi tiết tiến trình .....	71
Hình 14: Tập kết xuất bộ nhớ.....	73
Hình 15: Thư mục hệ điều hành được mở rộng trong GUI.....	74
Hình 16: GUI qui trình chi tiết .....	75
Hình 17: Option xuất dữ liệu.....	75
Hình 18: Network GUI .....	76
Hình 19: Danh sách các tệp đang mở .....	76
Hình 20: Nội dung của bộ nhớ tiến trình trong BinText .....	79
Hình 21: Chuỗi phiên bản được tìm thấy trong dd.exe.img với BinText .....	83

## Chapter 3: Windows Forensic Analysis

### 1. Windows Memory Analysis

#### 1.1. Giới thiệu

Trong Chương 1, chúng ta đã thảo luận về việc thu thập dữ liệu dễ bị mất khỏi hệ thống Windows khi đang hoạt động. Từ thứ tự các dữ liệu bị mất được liệt kê trong RFC 3227, đã chỉ ra rằng một trong những mục đầu tiên của dữ liệu bị mất, được thu thập trong các hoạt động phản hồi trực tiếp là nội dung của bộ nhớ vật lý, thường là do RAM. Mặc dù đã chỉ ra các chi tiết cụ thể của việc thu thập dữ liệu từ bộ nhớ bị đánh mất, chẳng hạn như kết nối mạng hoặc tiến trình đang chạy, vấn đề thu thập và phân tích đã được biết đến từ lâu và được thảo luận khá rộng rãi từ nhiều năm trước; vấn đề của thu thập và phân tích toàn bộ nội dung của bộ nhớ vật lý là một phát triển mới kể cả đối với sự phát triển vượt bậc ngày nay. Lĩnh vực nghiên cứu này đã được bắt đầu và mở rộng từ nhiều năm trước, bắt đầu từ mùa hè năm 2005, ít nhất là từ góc độ cộng đồng.

Câu hỏi quan trọng nhất cần được trả lời vào thời điểm này là Tại sao?

Tại sao bạn muốn thu thập nội dung của RAM? Việc này hữu ích như thế nào, nó quan trọng ra sao và bạn sẽ bỏ lỡ điều gì nếu không thu thập và phân tích nội dung của RAM? Cho đến bây giờ, một số nhà điều tra thu thập nội dung của RAM với hy vọng rằng sẽ không thể tìm thấy thứ gì đó trên ổ cứng và họ đã biết được đó chính là mật khẩu. Chương trình sẽ nhắc người dùng nhập mật khẩu, khi đó nếu hộp thoại biến mất khỏi màn hình, bộ nhớ chính là nơi có khả năng để tìm mật khẩu đó.

Các nhà phân tích phần mềm độc hại sẽ tìm đến bộ nhớ để xử lý các phần mềm độc hại bị mã hóa hoặc ẩn, bởi vì khi phần mềm độc hại được khởi chạy, nó sẽ được giải mã trong bộ nhớ. Càng ngày, Malware càng phức tạp, khiến cho việc

phân tích tĩnh, offline càng trở nên khó khăn hơn bao giờ hết. Tuy nhiên, nếu phần mềm độc hại được phép thực thi, nó sẽ tồn tại trong trạng thái đã được giải mã, điều này giúp việc phân tích phần mềm độc hại dễ dàng hơn. Cuối cùng, rootkit sẽ ẩn các tiến trình, các tệp, khóa Registry và thậm chí các kết nối mạng từ chế độ xem bằng các công cụ chúng ta thường sử dụng để thống kê, nhưng bằng cách phân tích nội dung của RAM, chúng ta có thể tìm thấy những gì bị ẩn. Chúng ta cũng có thể tìm thấy thông tin về các tiến trình đã thoát.

Năm 2008, Greg Hoglund (có lẽ được biết đến nhiều nhất khi viết rootkit khả thi đầu tiên cho các hệ thống Windows và rootkit.com, cũng như CEO của HBGary, Inc.) đã viết một bài báo có tiêu đề là “Giá trị của việc phân tích bộ nhớ vật lý đối với xử lý sự cố”. ([www.hbgary.com / resource.html](http://www.hbgary.com/resource.html)). Trong bài báo đó, Greg mô tả những gì có thể trích xuất từ nội dung của bộ nhớ vật lý và có lẽ quan trọng nhất là giá trị của thông tin đó đến việc giải quyết các vấn đề trong xử lý sự cố và phân tích máy tính.

## **1.2. Lược sử**

Trong quá khứ, khái niệm “phân tích” về dữ liệu bộ nhớ bao gồm việc chạy các chuỗi hoặc grep đối với tập tin hình ảnh, tìm kiếm mật khẩu, địa chỉ IP, địa chỉ email hoặc các chuỗi khác có thể giúp nhà phân tích điều tra. Hạn chế của phương pháp phân tích này là rất khó để gắn thông tin bạn tìm thấy vào một tiến trình riêng biệt. Phát hiện địa chỉ IP là một phần của tiến trình, hoặc nó thực sự được sử dụng bởi các tiến trình khác? Làm thế nào về nhận biết các từ trông giống như mật khẩu? Đây có phải là mật khẩu mà kẻ tấn công sử dụng để truy cập một Trojan trên hệ thống, hay nó là một phần của cuộc trò chuyện nhắn tin tức thời (IM)?

Có thể thực hiện một số loại phân tích về các bộ nhớ vật lý được đánh giá cao dựa trên danh sách đề cử của nhiều tổ chức đáng tin cậy. Những người khác

(chẳng hạn như tôi) đã nhận ra sự cần thiết của các công cụ và các frameworks đối với việc khôi phục bộ nhớ vật lý và phân tích nội dung của chúng.

Vào mùa hè năm 2005, Hội thảo Nghiên cứu Pháp lý Kỹ thuật số (DFRWS; [www.dfrws.org](http://www.dfrws.org)) đã đưa ra “thử thách phân tích bộ nhớ” nhằm thúc đẩy sự nghiên cứu, khám phá, phát triển các công cụ mới. Bất cứ ai cũng được mời tải xuống hai tập tin chứa các kho của bộ nhớ vật lý (các kho đã thu được bằng cách sử dụng một bản sao sửa đổi của dd.exe có sẵn trên bản phân phối Helix 1.6) và trả lời các câu hỏi dựa trên kịch bản được cung cấp trên Website. Chris Betz và bộ đôi George M. Garner, Jr. và Robert-Jan Mora được chọn là người chiến thắng chung cuộc của thử thách, họ cung cấp các bài viết minh họa xuất sắc về phương pháp này và thử nghiệm kết quả của các công cụ họ đã phát triển. Không may, những công cụ này đã không được công bố công khai.

Vào năm sau đó, những người khác tiếp tục nghiên cứu này hoặc nghiên cứu theo con đường riêng của họ. Andreas Schuster bắt đầu phát hành phiên bản tiếng Anh các phần nghiên cứu trên blog của anh ấy, với định dạng của cấu trúc EProcess và EThread từ các phiên bản khác nhau của Windows, bao gồm Windows 2000 và XP. Joe Stewart đã đăng một Perl script gọi là *pmodump.pl* như là một phần của Dự án Truman ([www.secureworks.com/research/tools/truman.html](http://www.secureworks.com/research/tools/truman.html)), cho phép bạn trích xuất dữ liệu được sử dụng bởi một tiến trình (quan trọng đối với phân tích phần mềm độc hại). Mariusz Burdach công bố thông tin liên quan đến phân tích bộ nhớ (ban đầu cho các hệ thống Linux nhưng sau đó là cho các hệ thống Windows) bao gồm một bài diễn thuyết tại hội nghị BlackHat Liên bang 2006. Jesse Kornblum có đưa ra một số kiến thức trong lĩnh vực phân tích bộ nhớ để xác định bản gốc hệ điều hành từ nội dung của bộ nhớ kết xuất. Mùa hè năm 2006, Tim Vidas, (<http://nucia.unomaha.edu/tvidas/>), một nghiên cứu viên cao cấp tại Đại học

Nebraska đã phát hành *procloc.pl*, một tập lệnh Perl để xác định vị trí các tiến trình trong RAM cũng như crash dumps.

Kể từ đó, lĩnh vực nghiên cứu liên quan đến thu thập và phân tích kết xuất bộ nhớ đã phát triển với những bước nhảy vọt, và trong nhiều lần, những thông số quan trọng đã tăng lên đáng kể. Cá nhân đáng chú ý nhất trong lĩnh vực phân tích bộ nhớ là Aaron Walters, đồng sáng tạo của FATKit (<http://4tphi.net/fatkit/>) và Volatility Frameworks ([https:// www.volatilesystems.com/default/volatility](https://www.volatilesystems.com/default/volatility)). Mặc dù các công cụ đã được phát hành cho phép thu thập nội dung của bộ nhớ vật lý từ các hệ thống Windows XP và Vista (được giải thích chi tiết trong chương này), Aaron và nhà đồng phát triển của mình, Nick L. Petroni Jr., vẫn tập trung chủ yếu cung cấp một framework để phân tích các kho bộ nhớ. Aaron và Nick có được sự hỗ trợ từ Brendan Dolan-Gavitt cùng những người khác ; đã có đóng góp đáng kể cho lĩnh vực phân tích bộ nhớ và đặc biệt là Volatility Framework.

Matthieu Suiche ([www.msuiche.net/](http://www.msuiche.net/)) đã đóng góp một chương trình nằm lấy được nội dung của bộ nhớ vật lý và cũng đã công bố những thông tin , công cụ này là công cụ để phân tích các tệp đóng băng của Windows (đã được tích hợp vào Volatility Framework). Andreas Schuster (<http://computer.forensikblog.de/en/>) đã tiếp tục có những đóng góp vào việc phân tích cú pháp bộ nhớ Windows, bao gồm cả việc phát hành một PT Downloader (sẽ thảo luận sau trong chương này) cho Vista vào tháng 11 năm 2008, được đưa vào như một phần trong PT Downloader của các tập lệnh Perl.

Ngoài các công cụ mã nguồn mở miễn phí để phân tích cú pháp bộ nhớ Windows, công ty bảo mật Mandiant (trang web của công ty là [www.mandiant.com](http://www.mandiant.com) và blog của công ty là <http://blog.mandiant.com/>) đã phát hành bộ nhớ *Memoryze* - công cụ thu thập và phân tích cú pháp, cùng với công cụ Audit Viewer để trình bày tốt hơn các kết quả của Memoryze.

Ngoài ra, HBGary ([www.hbgary.com/](http://www.hbgary.com/)) đã phát hành ứng dụng phân tích bộ nhớ của riêng họ có tên Responder, ứng dụng chuyên nghiệp trong lĩnh vực này. Trang web HBGary bao gồm rất nhiều thông tin về các công cụ, bao gồm các video hướng dẫn sử dụng .

### 1.3. Collecting Process Memory

Trong suốt tiến trình điều tra, bạn có thể chỉ quan tâm đến các tiến trình cụ thể thay vì danh sách tất cả các tiến trình và có thể quan tâm nhiều hơn đến không chỉ là nội dung có sẵn của bộ nhớ tiến trình có trong kho chứa file của RAM. Ví dụ: bạn có thể đã nhanh chóng xác định được các tiến trình không yêu cầu điều tra mở rộng bổ sung. Có nhiều cách để thu thập tất cả bộ nhớ được sử dụng bởi một tiến trình, không chỉ những gì trong bộ nhớ vật lý, mà cả những gì trong bộ nhớ ảo hoặc tập tin trang.

Để làm điều này, có thể sử dụng một vài công cụ có sẵn như là *pmdump.exe* ([www.ntsecurity.nu/tools/pmdump/](http://www.ntsecurity.nu/tools/pmdump/)), được viết bởi Arne Vidstrom .Tuy nhiên, như trang web NTSecurity.nu tuyên bố, *pmdump.exe* cho phép bạn kết xuất nội dung của bộ nhớ xử lý mà không dừng tiến trình. Như đã thảo luận trước đó, điều này cho phép tiến trình tiếp tục và nội dung của bộ nhớ thay đổi khi nó được ghi vào một tệp, do đó sẽ tạo ra một “dấu vết” của tiến trình bộ nhớ. Ngoài ra, *pmdump.exe* không tạo tệp đầu ra có thể được phân tích bằng Debugging Tools(Debugging Tools) của Microsoft.

Tobias Klein đã đưa ra một phương pháp khác để loại bỏ nội dung của bộ nhớ tiến trình dưới dạng công cụ miễn phí (mặc dù không phải là nguồn mở) Process Dumper (có sẵn trong cả hai phiên bản Linux và Windows từ [www.trapkit.de/research/forensic/pd/index.html](http://www.trapkit.de/research/forensic/pd/index.html)). Process Dumper (*pd.exe*) kết xuất toàn bộ không gian tiến trình cùng với siêu dữ liệu bổ sung và môi trường xử



lý vào bảng điều khiển (STDOUT) để đầu ra có thể được chuyển hướng đến một tệp hoặc socket (thông qua netcat hoặc các công cụ khác; xem Chương 1 để biết thêm kiến thức về một trong số những công cụ đó). Việc xem xét tài liệu mà Tobias cung cấp cho *pd.exe* không cho thấy dấu hiệu nào về việc tiến trình được gỡ lỗi, tạm dừng hoặc đóng băng trước khi tiến trình bị thay đổi. Tobias cũng cung cấp giao diện người dùng đồ họa GUI (GUI) tiện lợi cho việc phân tích siêu dữ liệu và nội dung bộ nhớ được thu thập bởi Process Dumper. Các công cụ này dường như là một phần mở rộng của Tobias Time, hoạt động để trích xuất các khóa riêng RSA và chứng chỉ từ bộ nhớ tiến trình ([www.trapkit.de/research/sslkeyfinder/index.html](http://www.trapkit.de/research/sslkeyfinder/index.html)).

Một công cụ khác có sẵn và được đề xuất bởi một số nguồn chính là *userdump.exe*, từ Microsoft. *Userdump.exe* sẽ cho phép bạn kết xuất bất kỳ tiến trình nào một cách nhanh chóng mà không cần đính kèm trình gỡ lỗi và không chấm dứt tiến trình sau khi kết xuất. Ngoài ra, tệp kết xuất được tạo bởi *userdump.exe* có thể được đọc bởi Debugging Tools của Microsoft. Tuy nhiên, *userdump.exe* yêu cầu trình điều khiển được cài đặt để nó hoạt động, và tùy thuộc vào tình huống, đây có thể không phải là điều bạn muốn. Dựa trên các cuộc trò chuyện với Robert Hensing, trước đây thuộc nhóm Bảo mật PSS của Microsoft, phương pháp loại bỏ bộ nhớ ưa thích được sử dụng *adplus.vbs*, hỗ trợ các thư viện liên kết động hoặc DLL) có thể ghi vào đĩa CD (*adplus.vbs* sử dụng máy chủ lưu trữ kịch bản Windows Phiên bản 5.6, còn được gọi là *cscript.exe*, được cài đặt trên hầu hết các hệ thống) và được sử dụng để kết xuất các tiến trình vào ổ đĩa chung hoặc đến bộ lưu trữ được kết nối USB. Khi các vùng chứa đã hoàn thành, bạn có thể sử dụng MS Debugging tool có sẵn để phân tích các tệp kết xuất. Ngoài ra, bạn có thể sử dụng các công cụ khác, chẳng hạn như BinText, để trích xuất ASCII, Unicode và chuỗi tài nguyên từ tệp kết xuất. Ngoài ra, sau khi đã bỏ tiến trình, bạn vẫn có thể sử dụng các công cụ khác (như những công cụ được thảo luận trong

Chương 1) để thu thập thông tin bổ sung về tiến trình từ hệ thống đang chạy, có thể cung cấp cái nhìn nhanh hơn về chi tiết của tiến trình. File .exe (có sẵn từ <http://technet.microsoft.com/en-us/sysinternals/bb896655.aspx> , có quyền Run as administrator) sẽ cung cấp danh sách các thẻ điều khiển (đối với tệp, thư mục, v.v.) đã được mở bởi tiến trình và listdlls.exe (Phiên bản 2.25 tại thời điểm viết bài này, có sẵn từ <http://technet.microsoft.com/en-us/sysinternals/bb896656.aspx>) sẽ hiển thị cho bạn đường dẫn đầy đủ và version của các mô-đun.

Trợ giúp mở rộng có sẵn adplus.vbs, không chỉ trong tệp trợ giúp Debugging ToolsMS mà còn trong bài viết 286350 của Support Microsoft (<http://support.microsoft.com/kb/286350>). Bạn có thể sử dụng adplus.vbs để treo tiến trình trong khi nó đang bị hủy (nghĩa là tạm dừng nó, kết xuất nó và sau đó tiếp tục tiến trình) hoặc làm hỏng tiến trình . Để chạy adplus.vbs trong chế độ treo đối với một tiến trình, bạn sẽ sử dụng dòng lệnh sau:

***D: \ debug> cscript adplus.vbs -quiet -hang <PID>***

Lệnh này sẽ tạo một loạt các tệp trong thư mục gỡ lỗi trong thư mục con được mở đầu bằng tên *Hang\_mode\_* bao gồm ngày và giờ (Bạn có thể thay đổi vị trí output được ghi bằng cách sử dụng khóa chuyển đổi.) Bạn sẽ thấy tệp báo cáo *adplus.vbs*, tệp kết xuất cho tiến trình (có thể có nhiều tiến trình được chỉ định bằng nhiều mục nhập của pep), một danh sách tiến trình (được tạo theo mặc định bằng cách sử dụng *tlist.exe*; bạn có thể tắt tính năng này bằng cách sử dụng khóa chuyển đổi).

(DLL) được sử dụng bởi tiến trình. Mặc dù tất cả thông tin được thu thập về các tiến trình sử dụng adplus.vbs đều cực kỳ hữu ích trong tiến trình điều tra, nhưng bạn chỉ có thể sử dụng công cụ này trên các tiến trình hiển thị thông qua API. Nếu một tiến trình không hiển thị (giả sử, nếu nó bị ẩn bởi rootkit), bạn không thể sử dụng các công cụ này để thu thập thông tin về tiến trình.

Bạn có thể sử dụng lệnh *memdump* động để kết xuất bộ nhớ có thể định địa chỉ cho một tiến trình từ kết xuất bộ nhớ Windows XP, như sau:

Vận chuyển với các công cụ gỡ lỗi, vì phương pháp này gắn trình gỡ lỗi vào tiến trình và tạm dừng tiến trình để kết xuất nó. *Adplus* là viết tắt của *Autodumplus* và ban đầu là được viết bởi Robert (tài liệu cho kịch bản nói rằng các phiên bản 1 đến 5 được viết bởi Robert và kể từ Phiên bản 6, Israel Burman đã tiếp quản). Tập trợ giúp (debugger.chm) cho MS Debugging tool chứa rất nhiều thông tin về tập lệnh cũng như Debugging Toolscdb.exe mà nó sử dụng để kết xuất các tiến trình được chỉ định. MS Debugging tool không yêu cầu phải cài đặt trình điều khiển bổ sung và chúng có thể được chạy từ đĩa CD.

#### **1.4. Dumping Physical Memory**

Vậy, làm thế nào để bạn thu thập đầy đủ nội dung của bộ nhớ vật lý, thay vì chỉ bỏ một tiến trình? Một số phương pháp có sẵn, mỗi phương pháp đều có điểm mạnh và điểm yếu.

Mục tiêu của chương này là cung cấp sự hiểu biết về các tùy chọn khác nhau có sẵn cũng như các khía cạnh kỹ thuật liên quan đến từng tùy chọn. Bằng cách này, với tư cách là người trả lời hoặc điều tra viên đầu tiên, bạn sẽ chọn phương án nào phù hợp nhất, đưa nhu cầu kinh doanh của khách hàng (hoặc nạn nhân) vào cùng với các mối quan tâm về cơ sở hạ tầng.

Trong dòng lệnh trước, kích thước khối (giá trị bs) được đặt thành 4 kilobyte (KB) hoặc 4096 byte, vì đây là kích thước mặc định cho các trang trong bộ nhớ. Do đó, lệnh yêu cầu dd.exe lấy một trang mỗi lần. Phiên bản dd.exe này cũng cho phép nén và băm mật mã cho nội dung. Do tính chất dễ thay đổi của RAM (tức là, nó tiếp tục thay đổi trong suốt tiến trình), tuy nhiên, không nên băm nó cho đến khi được ghi từ đĩa. Nếu người dùng bộ nhớ hình ảnh hai lần, thậm chí với độ trễ

nhỏ, nội dung của RAM và các giá trị băm tiếp theo sẽ khác nhau. Trong trường hợp này, chỉ nên sử dụng hàm băm mật mã để bảo đảm tính toàn vẹn của kết xuất bộ nhớ được thu thập.

#### 1.4.1. Nigilant32

Các công cụ khác sử dụng một tiến trình tương tự như dd.exe để ghi lại nội dung RAM. ([www.agilerm.net/publications\\_4.html](http://www.agilerm.net/publications_4.html)), từ Matt Shannon của Quản lý rủi ro nhanh, sử dụng giao diện đồ họa để cho phép bộ phản hồi thu được nội dung của bộ nhớ vật lý.

Hình 1 minh họa một phần của GUI Nigilant32, hiển thị tùy chọn cho hình ảnh bộ nhớ vật lý.



Hình 1: Một phần GUI Nigilant32

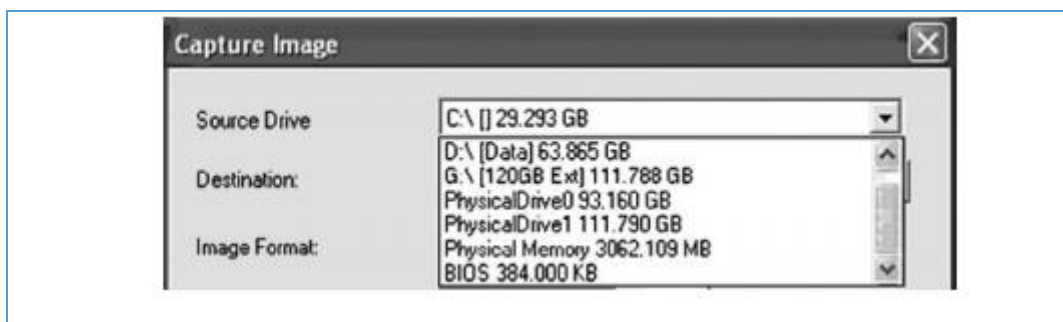
Nigilant32 có lợi thế về GUI, là một công cụ thân thiện đối với người dùng. Bạn có thể triển khai Nigilant32 trên ổ đĩa CD hoặc USB cùng với các công cụ khác và sử dụng nó để nhanh chóng thu thập nội dung của bộ nhớ vật lý (chủ yếu từ

Hệ thống Windows XP). Cũng như dd.exe và các công cụ khác, Nigilant32 yêu cầu quyền truy cập cục bộ vào hệ thống mà từ đó bộ phản hồi muốn kết xuất bộ nhớ. Bạn cũng có thể triển khai Nigilant32 trước khi xảy ra sự cố thực tế và

người phản hồi có thể chạy công cụ cục bộ và chuyển bộ nhớ vật lý vào ổ đĩa , chia sẻ mạng hoặc truy cập hệ thống từ xa (thông qua các ứng dụng như VNC hoặc Remote Desktop Client) và thực hiện kết xuất bộ nhớ.

#### 1.4.2. ProDiscover

ProDiscover IR (Phiên bản 4.8 đã được sử dụng để viết cuốn sách này và Phiên bản 5.0 được phát hành vào mùa hè 2008) cũng cho phép nhà điều tra thu thập nội dung của bộ nhớ vật lý (cũng như BIOS hệ thống) ; thông qua một applet máy chủ từ xa có thể được phân phối đến hệ thống thông qua phương tiện lưu trữ di động (CD, ổ ngón tay cái, v.v.) hoặc qua mạng. Hình 2 minh họa giao diện người dùng cho khả năng này.



Hình 2: Trích đoạn Hộp thoại Chụp từ ProDiscover IR

#### 1.4.3. KnTDD

Tuy nhiên, vấn đề với việc sử dụng các công cụ như dd.exe hoặc Nigilant32 là do Windows 2003 SP1, quyền truy cập vào đối tượng \ Device \ PhysMemory đã bị hạn chế từ chế độ người dùng . Đó là, chỉ có trình điều khiển kernel được phép truy cập đối tượng này. Do đó, các công cụ như dd.exe, Nigilant32 và ProDiscover (như đã đề cập trước đây, phiên bản Phản hồi sự cố của ProDiscover bao gồm một servlet cho phép nhà phân tích truy cập bộ nhớ vật lý trên hệ thống Windows XP từ xa) sẽ không cho phép bạn thu thập nội dung của bộ nhớ vật lý từ Windows 2003 SP1 và các hệ thống mới hơn, bao gồm cả Vista. Để giải quyết vấn đề này, George

M. Garner, Jr. (a.k.a. GMG Systems) đã sản xuất một tiện ích mới có tên KnTDD, một phần của bộ tiện ích KnTTools. Theo giấy phép cho KnTTools, các tiện ích có sẵn để bán riêng cho nhân viên thực thi pháp luật và các chuyên gia bảo mật thực sự.

KnTDD bao gồm các khả năng sau:

- Có được nội dung của bộ nhớ vật lý bằng nhiều phương thức, bao gồm cả thông qua đối tượng \ Device \ PhysMemory
- Chạy trên các hệ điều hành Microsoft từ Windows 2000 đến Vista, bao gồm các phiên bản AMD64 của các hệ điều hành
- Có thể chuyển đổi bộ nhớ thô Hình ảnh trực tiếp sang định dạng kết xuất lỗi của Microsoft để có thể phân tích dữ liệu kết quả bằng Debugging Tools của Microsoft
- Có thể mua thiết bị lưu trữ cục bộ (USB, FireWire) cũng như qua mạng bằng Giao thức điều khiển truyền / Giao thức Internet (TCP / IP)
- Được thiết kế đặc biệt để phân tích, với nhật ký kiểm tra và kiểm tra tính toàn vẹn của mật mã
- Phiên bản KnTTools Enterprise bao gồm các khả năng sau:
- Mã hóa hàng loạt đầu ra bằng các chứng chỉ X.509, AES-256 (mặc định) và hạ cấp xuống 3DES trên Windows 2000
- Thu thập bộ nhớ bằng dịch vụ KnTDDSvc

Mô-đun triển khai từ xa có thể triển khai dịch vụ KnTDDSvc bằng cách đẩy nó sang chia sẻ quản trị viên từ xa hoặc kéo kéo nó từ máy chủ Web qua Lớp công bảo mật (SSL), với xác minh mật mã của các nhị phân trước khi chúng Thực thi

Một trong những khía cạnh của việc sử dụng dd.exe và các công cụ như giống như vậy mà bạn cần ghi nhớ là Nguyên tắc trao đổi Locard. Để sử dụng các

công cụ này cho thu thập nội dung của RAM, chúng phải được tải vào RAM tiến trình đang chạy. Điều này có nghĩa là không gian bộ nhớ được sử dụng và các tiến trình khác có thể có các trang được ghi ra tệp.

Một khía cạnh khác của các công cụ này cần lưu ý là chúng không đóng băng trạng thái của hệ thống, như xảy ra khi một sự cố được tạo ra. Điều này có nghĩa là trong khi công cụ đang đọc thông qua nội dung của RAM, vì trang thứ mười ba đang được đọc trang thứ mười một có thể thay đổi khi tiến trình sử dụng trang đó tiếp tục chạy. Lượng thời gian cuối cùng cần để hoàn thành kho phụ thuộc vào các yếu tố như tốc độ xử lý và tốc độ của các Bus và đĩa I / O.

Những thay đổi này xảy ra trong khoảng thời gian giới hạn có đủ ảnh hưởng đến kết quả phân tích của bạn không? Trong hầu hết các điều kiện phản ứng sự cố, các công cụ như dd.exe có thể là phương pháp tốt nhất để truy xuất nội dung của bộ nhớ vật lý, đặc biệt là từ Windows 2000 và XP hệ thống. Các công cụ như vậy không yêu cầu phải gỡ bỏ hệ thống, cũng như không giới hạn cách thức và nơi ghi nội dung của bộ nhớ vật lý (ví dụ: sử dụng netcat, bạn có thể viết nội dung của RAM vượt qua ổ cắm sang hệ thống khác chứ không phải ổ cứng cục bộ).

Tiện ích của KnTTools là hệ thống vẫn đang chạy khi nội dung của bộ nhớ vật lý được lấy. Điều này có nghĩa là không chỉ các trang bộ nhớ được tiêu thụ đơn giản bằng cách sử dụng các tiện ích (tức là, hình ảnh thực thi được đọc và được tải vào bộ nhớ), nhưng khi công cụ liệt kê qua bộ nhớ, trang đã được đọc có thể thay đổi. Đó là, trạng thái của hệ thống và bộ nhớ của nó không bị đóng băng trong thời gian, như trường hợp có được một hình ảnh phân tích của một ổ đĩa cứng thông qua các phương pháp phân tích truyền thống.

#### 1.4.4. MDD

Đầu năm 2008 đã chứng kiến sự phát hành của một số công cụ mới để thu thập nội dung của bộ nhớ vật lý từ các hệ thống Windows, đặc biệt là Windows 2003 SP1 và Vista. May mắn thay, các công cụ này hoạt động tốt như nhau trên các hệ thống Windows XP, khiến chúng trở nên phổ quát hơn các công cụ có sẵn trước đây (ví dụ: dd.exe, Nigilant32, v.v.). Đáng chú ý là mdd.exe được phát hành ra công chúng bởi ManTech International([www.mantech.com/msma/MDD.asp](http://www.mantech.com/msma/MDD.asp)). Mdd.exe là một công cụ giao diện dòng lệnh đơn giản và đơn giản (CLI), cho phép người trả lời bỏ nội dung của bộ nhớ vật lý với một dòng lệnh đơn giản:

***E:\response>mdd.exe -o F:\system1\memory.dmp***

Dòng lệnh trên minh họa một ví dụ về mdd.exe chạy từ CD (E: \), gửi tệp đầu ra (thông qua bộ chuyển đổi) đến ổ cứng ngoài USB (F: \). Bạn cũng có thể chạy mdd.exe từ một tệp, như tôi đã mô tả trong Chương 1, sử dụng một dòng lệnh như:

***mdd.exe -o % 1 \ mdd-o.img***

Ngoài ra, bạn có thể sử dụng các biến có sẵn từ môi trường Windows để trực tiếp phân tách dữ liệu dễ mất đi khi thu thập bằng cách thêm tên của hệ thống:

***mdd.exe -o % 1 \% ComputerName% -mdd-o.img***

Khi mdd.exe hoàn thành kết xuất bộ nhớ, nó sẽ hiển thị tổng kiểm MD5 cho tệp kết xuất kết quả: mất 108 giây để ghi

***MD5 là: 6fe975ee3ab878211d3be3279e948649***

Nhà phân tích có thể lưu thông tin này và sử dụng nó để đảm bảo tính toàn vẹn của bộ nhớ.

Mdd.exe không chỉ đơn giản để sử dụng và triển khai, mà cả tìm ra đầu ra của công cụ là gì (được gọi là tệp kết xuất bộ nhớ thô), kiểu dd, tương tự như những gì đạt được bằng cách sử dụng khác các công cụ (dd.exe, VMware, v.v.). Tuy nhiên, trước khi sử dụng mdd.exe, bạn nên lưu ý rằng, theo trang web



ManTech International cho công cụ này, mdd.exe không thể thu thập hơn 4GB RAM. Trên các hệ thống có RAM 8GB trở lên, một số người dùng đã báo cáo sự cố hệ thống, vì vậy hãy tính đến điều này khi xem xét liệu có nên thu thập nội dung của RAM từ hệ thống hay không.

### **Chú ý**

Nếu bạn có ý định thu thập nội dung của RAM từ hệ thống Windows trực tiếp như một phần của phương pháp phản hồi bằng tệp, tôi khuyên bạn nên thu thập nội dung của RAM trước. Việc này sẽ dọn dẹp một cách tốt nhất có thể, đặc biệt là nếu đã bao gồm các bên thứ ba khác (tlist.exe, tcpvcon.exe) và các công cụ Windows gốc (netstat.exe) trong tệp đó.

#### *1.4.5. Win32dd*

Matthieu Suiche đã phát hành công cụ của riêng mình để bỏ nội dung của bộ nhớ vật lý, được gọi là win32dd (<http://win32dd.msuiche.net/>). Win32dd được mô tả là vùng đất hạt nhân miễn phí và công cụ mã nguồn mở 100%; điều này có nghĩa là giống như mdd.exe, win32dd.exe là miễn phí, nhưng không giống như công cụ ManTech, win32dd.exe là nguồn mở. Win32dd.exe có một số tính năng bổ sung, bao gồm khả năng tạo ra một bản đồ lỗi tương thích với WinDbg, bị lỗi (tương tự như bản sửa lỗi Windows) sau đó bạn có thể phân tích bằng Debugging Tools của Microsoft ([www.microsoft.com/whdc/ DevTools / Gỡ lỗi / default.msp](http://www.microsoft.com/whdc/DevTools/Gỡ%20lỗi/default.msp)). Như với mdd.exe và các công cụ CLI khác, win32dd.exe có thể được bao gồm trong các tệp bố. Phiên bản 1.2.1.20090106 của win32dd.exe đã có sẵn tại thời điểm chương này được viết. Bạn có thể xem thông tin cú pháp có sẵn cho win32dd.exe bằng cách sử dụng dòng lệnh sau:

***D:\tools\win32dd>win32dd -h***

Win32dd - v1.2.1.20090106 - Kernel land physical memory acquisition  
Copyright (c) 2007 - 2009, Matthieu Suiche

Copyright (c) 2008 - 2009, MoonSols

Usage: win32dd [option] [output path]

Option:

- r Create a raw memory dump/snapshot. (default)
- l Level for the mapping (with -r option only).
  - l 0 Open \\Device\\PhysicalMemory device (default).
  - l 1 Use Kernel API MmMapIoSpace()
- d Create a Microsoft full memory dump file (WinDbg compliant).
- t Type of MSFT dump file (with -d option only).
  - t 0 Original MmPhysicalMemoryBlock, like BSOD. (default).
  - t 1 MmPhysicalMemoryBlock (with PFN 0).
- h Display this help.

Sample:

Usage: win32dd -d physmem.dmp

Usage: win32dd -l 1 -r C:\dump\physmem.bin

Như bạn có thể thấy, win32dd.exe có một số tùy chọn có sẵn để bỏ nội dung của bộ nhớ vật lý. Để tạo tệp kết xuất bộ nhớ thô, kiểu dd, bạn có thể sử dụng như sau

dòng lệnh:

***D:\tools\win32dd>win32dd -l 0 -r d:\tools\memtest\win32dd-l0-xp.img***

Bao gồm các tùy chọn dòng lệnh (chẳng hạn như 0) trong tên của tệp đầu ra đóng vai trò là mô-đun của tài liệu bổ sung cho nhà phân tích, để xác định các dòng lệnh được sử dụng. Cũng như các công cụ CLI khác, bao gồm một lệnh như vậy trong tệp rất đơn giản và dễ hiểu (bên cạnh việc tự ghi lại tài liệu).

#### *1.4.6. Memoryze*

Công ty tư vấn Mandiant đã phát hành công cụ phân tích và thu thập bộ nhớ của riêng mình, được gọi là Memoryze. Memoryze có nguồn gốc từ sản phẩm Intelligent Responder Mandiant (MIR) và đã được cung cấp miễn phí từ trang web của Mandiant, cùng với các ví dụ về cách chạy công cụ. Khi bạn đã tải xuống tệp Trình cài đặt Microsoft (MSI) vào hệ thống của mình và cài đặt nó, bạn có thể chạy Memoryze để thu thập kết xuất bộ nhớ thông qua tệp memorydd.bat bằng cách nhập lệnh sau:

***D:\Mandiant\memorydd.bat***

Điều này sẽ tạo ra một tệp kết xuất bộ nhớ trong một cấu trúc thư mục hiện tại; chạy lệnh như trên hệ thống: Audits\WINTERMUTE\20090103003442\, và trong thư mục cuối cùng mà nó tạo ra là một số tệp XML và một tệp hình ảnh bộ nhớ. Chạy tệp với khóa chuyển đổi outout sẽ cho phép bạn xác định cấu hình vị trí cho tệp kết xuất.

Ngoài ra, khi chạy trên một hệ thống, Memoryze sẽ báo cáo sử dụng tệp trang, kết hợp nội dung bộ nhớ từ tệp trang vào tiến trình thu thập. Điều này cho phép thu thập dữ liệu đầy đủ hơn, vì nội dung bộ nhớ đã được hoán đổi thành tệp tin trang hiện có sẵn trong tiến trình phân tích. Ngoài ra, Memoryze Phiên bản 1.3.0 (công bố ngày 9 tháng 2 năm 2009) có khả năng kết xuất bộ nhớ có thể truy cập thông qua F-Feedback, mà chúng tôi sẽ thảo luận sau trong chương này.

#### *1.4.7. Winen*

Phần mềm hướng dẫn cũng phát hành công cụ riêng để thu thập nội dung của bộ nhớ vật lý, được gọi là winen.exe. Giống như một số công cụ khác, winen.exe là một công cụ CLI, nhưng không giống như các công cụ khác, kết xuất bộ nhớ không được thu thập ở định dạng thô, kiểu dd; thay vào đó, nó được thu

thập ở cùng định dạng hình ảnh độc quyền được sử dụng bởi công cụ thu thập hình ảnh EnCase, thường được gọi là Định dạng nhân chứng chuyên gia (hay gọi tắt là EWF). Hầu hết các công cụ phân tích có sẵn đều yêu cầu kết xuất bộ nhớ ở định dạng thô, kiểu dd, do đó bộ nhớ kết xuất thu thập bằng winen.exe phải được chuyển đổi sang định dạng thô trước khi được phân tích cú pháp. Như Lance Mueller chỉ ra trong blog của mình ([www.forensickb.com/2008/06/new-version-of-encaseincludes-stand.html](http://www.forensickb.com/2008/06/new-version-of-encaseincludes-stand.html)), bạn có thể chạy winen.exe bằng cách cung cấp các tùy chọn khác nhau tại bảng điều khiển (ví dụ: nhập giành chiến thắng tại dấu nhắc lệnh và sau đó cung cấp câu trả lời cho các truy vấn) hoặc bằng cách cung cấp đường dẫn đến tệp cấu hình với thông tin cần thiết thông qua khóa chuyển đổi. Winen.exe có tổng cộng sáu tùy chọn bắt buộc, các cài đặt có thể được cung cấp qua dòng lệnh hoặc trong tệp cấu hình: đường dẫn đến (các) tệp đầu ra, mức nén, tên của người kiểm tra, tên bằng chứng, số trường hợp, và số bằng chứng. Một tệp cấu hình ví dụ được bao gồm trong thư mục chính khi bạn tải xuống và cài đặt EnCase. Richard McQuown cung cấp thêm thông tin về việc sử dụng winen.exe trong blog ForensicZone của anh ấy (<http://forensiczone.blogspot.com/2008/06/winenexe-ram-imagingtool-included-in.html>), cũng như liên kết đến một tệp hướng dẫn sử dụng cho winen.exe (bạn phải có quyền truy cập vào Diễn đàn người dùng EnCase và đăng nhập để lấy tài liệu PDF tại <https://support.guidancesoftware.com/forum/doads.php?do=file&id=478>) Sau đó, bạn có thể chuyển đổi kết xuất bộ nhớ kết quả thành định dạng thô bằng cách mở tệp kết xuất bộ nhớ trong FTK Imager và chọn Tạo hình ảnh đĩa từ menu (<http://windowsir.blogspot.com/2008/06/memory-collection-and-analysis-part-ii.html>) hoặc bằng cách mở kết xuất bộ nhớ trong EnCase và chọn Sao chép / Hủy bỏ từ Menu EnCase. Guidance cũng có phiên bản winen.exe dành cho phiên bản 64 bit của hệ điều hành Windows, được gọi là winen64.exe. Giống như phiên bản

32 bit của công cụ, winen64.exe cho phép bộ phản hồi kết xuất nội dung của bộ nhớ vật lý trong tệp kết xuất định dạng EWF.

#### 1.4.8. *Fastdump*

Công ty tư vấn HBGary đã phát hành một bản sao của công cụ của mình để thu thập nội dung của bộ nhớ vật lý, được gọi là fastdump ([www.hbgy.com/doad\\_fastdump.html](http://www.hbgy.com/doad_fastdump.html)). Mặc dù nhanh chóng .exe (Phiên bản 1.2 có sẵn tại thời điểm viết bài này) là miễn phí, như với Nigilant32 và một số công cụ có sẵn khác, nó không thể thu thập nội dung bộ nhớ từ Windows 2003 SP1 và các hệ thống mới hơn, bao gồm cả Vista. Bạn chỉ có thể sử dụng công cụ này để thu thập nội dung của bộ nhớ vật lý từ các hệ thống Windows XP và Windows 2003 (không cài đặt gói dịch vụ). Để giải quyết vấn đề này, HBGary cũng đã phát hành một phiên bản thương mại của công cụ, được gọi là FastDump Pro (fdpro.exe), có sẵn từ cùng một trang web là fastdump.exe, mặc dù phải trả phí. Phiên bản thương mại hỗ trợ tất cả các phiên bản Windows, bao gồm 32 và 64 bit các phiên bản và cũng sẽ báo cáo bộ nhớ từ các hệ thống có hơn 4GB RAM. Bên cạnh việc có thể kết xuất hơn 4GB bộ nhớ vật lý, FastDump Pro còn có một số điểm khác biệt và ưu điểm khác so với các công cụ khác. Nhập fdpro tại dấu nhắc lệnh sẽ hiển thị thông tin cú pháp cho công cụ:

```
D:\HBGary\bin\FastDump>fdpro
-= FDPro v1.3.0.377 by HBGary, Inc -=
***** Usage Help *****
```

Như bạn có thể thấy, fdpro.exe về cơ bản có hai chế độ có thể sử dụng. Chế độ đầu tiên là kết xuất nội dung của bộ nhớ vật lý ở định dạng thô, kiểu dd thông thường, sử dụng trình điều khiển để truy cập bộ nhớ vật lý trên các phiên bản Windows yêu cầu (ví dụ: Windows 2003 SP1, Vista và sau này) hoặc công cụ chuyển đổi củannodriver để thực hiện việc mua lại bộ nhớ kiểu cũ.

Fdpro.exe cũng có định dạng kiểu .hpak, được mô tả như sau:

Định dạng HPAK là một định dạng độc quyền của HBGary, có khả năng với một số tính năng chính, cụ thể là khả năng lưu trữ và lưu trữ RAM và Pagefile trong một kho lưu trữ duy nhất. Định dạng HPAK cũng hỗ trợ nén bằng định dạng gzip. Điều này hữu ích trong các trường hợp không gian trên thiết bị / hệ thống thu thập bị giới hạn.

Một trong những hạn chế của hầu hết các công cụ có sẵn để loại bỏ bộ nhớ vật lý là chúng chỉ cho phép truy cập vào bộ nhớ vật lý và không kết hợp tệp trang. Fdpro.exe không chỉ cung cấp quyền truy cập vào một loạt các hệ điều hành Windows (bao gồm cả Phiên bản 64 bit) và dung lượng bộ nhớ (tức là lớn hơn 4GB), nhưng cũng như (như với công cụ Memoryze của Mandiant), công cụ sẽ kết hợp tệp trang trong tiến trình thu thập, sau đó cho phép dữ liệu đó được tích hợp vào tiến trình phân tích, cũng vậy (chúng ta sẽ thảo luận về Sản phẩm Phản hồi HBGary sau này trong chương này). Như bạn sẽ thấy sau trong chương này, nội dung của bộ nhớ đã được hoán đổi thành tệp trang có thể chứa thông tin phù hợp với phản hồi và phân tích của bạn.

#### *1.4.9. F-Response*

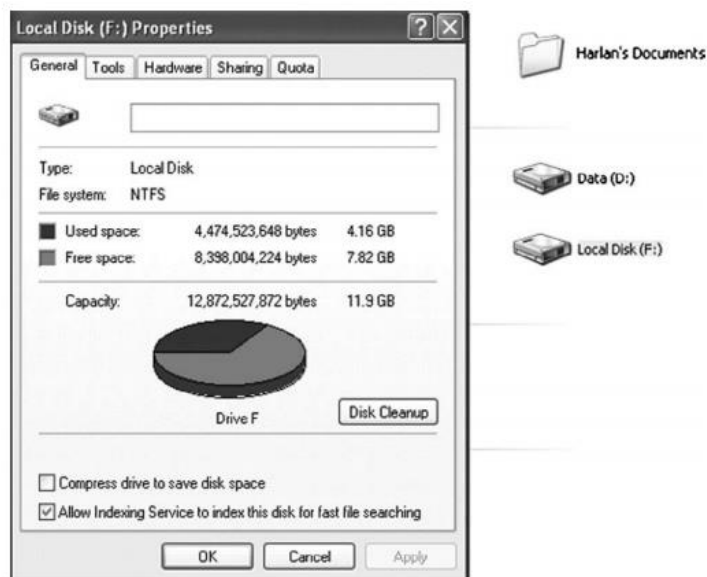
Cuối cùng, năm 2008 đã báo trước sự xuất hiện của một thời đại mới trong phản ứng sự cố với bản phát hành F-Feedback của Matt Shannon. F-Feedback là một khung công tác không xác định công cụ mua lại sử dụng giao thức iSCSI để cung cấp quyền truy cập thô vào các đĩa qua mạng. Matt đã thiết kế và đã viết F-Feedback để truy cập ở chế độ chỉ đọc; hoạt động ghi vào đĩa từ xa. Ba phiên bản của F - Feedback (Bộ công cụ, Chuyên gia tư vấn và Doanh nghiệp) đều được triển khai khác nhau, nhưng tất cả đều cho phép bạn nhìn thấy các ổ đĩa trên điều khiển từ xa các hệ thống (ví dụ: trong một phòng khác, trên một tầng khác hoặc thậm chí

trong một tòa nhà khác) như các ổ đĩa được gắn trên hệ thống của riêng bạn. F-Feedback là công cụ thu nhận công cụ theo nghĩa là một khi bạn đã kết nối với hệ thống từ xa và có thể gặp được các ổ đĩa, bạn có thể sử dụng bất kỳ công cụ nào bạn muốn (dd.exe, ProDiscover, FTK Imager, v.v.) để có được hình ảnh của ổ đĩa đó. Bạn có thể triển khai F-Feedback trên các hệ thống Mac OS X, Linux và Windows và trên các hệ thống Windows, nó có thêm lợi ích là cung cấp quyền truy cập từ xa, chỉ đọc vào bộ nhớ vật lý.

Tại Hội nghị phân tích Sans vào tháng 10 năm 2008, trong bài trình bày với Aaron Walters, Matt đã công bố phát hành F-Feedback 2.03, cung cấp quyền truy cập từ xa, thời gian thực, chỉ đọc vào bộ nhớ vật lý của hệ thống Windows, cho tất cả các phiên bản của Windows, bao gồm XP và Vista. Khi kết nối với hệ thống từ xa được thiết lập, bộ phản hồi có thể sử dụng các công cụ như dd.exe hoặc FTK Imager để có được một bản sao của bộ nhớ vật lý.

F-Response's capability (Phản hồi phiên bản 2.06 đã được sử dụng trong ví dụ này) để lấy bản sao bộ nhớ vật lý từ hệ thống Windows từ xa có ý nghĩa rất lớn đối với phản ứng sự cố, đặc biệt là khi triển khai Phiên bản doanh nghiệp. Matt có một số video được liên kết với blog F-Feedback ([www.fresponse.com/index.php?option=com\\_content&task=blogsection&id=4&Itemid=9](http://www.fresponse.com/index.php?option=com_content&task=blogsection&id=4&Itemid=9)) thể hiện việc sử dụng và các khía cạnh của công cụ Phản hồi F, chẳng hạn như minh họa cách lấy dữ liệu cụ thể, ví dụ, bằng cách truy cập máy chủ Microsoft Exchange trực tiếp.

Tại Hội nghị thượng đỉnh phân tích Sans vào tháng 10 năm 2008, Matt đã thông báo rằng F-Feedback sẽ cung cấp quyền truy cập từ xa, chỉ đọc vào các ổ đĩa trên hệ thống Windows và vào bộ nhớ vật lý. Khi kết nối thành công, các ổ đĩa từ xa sẽ xuất hiện trên hệ thống hồi đáp với biểu tượng ổ đĩa quen thuộc, như Hình 3 minh họa



Hình 3: Ổ đĩa hệ thống Windows từ xa được kết nối với F: \

## Tools & Traps...

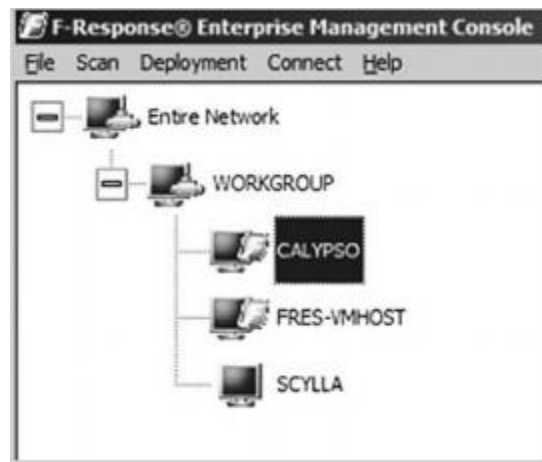
### F-Response Enterprise Edition (EE)

Bạn có thể triển khai các F-Feedback Enterprise Edition (EE) theo cách chủ động. Việc triển khai các tác nhân EE trước thời hạn có thể tăng tốc đáng kể thời gian đáp ứng, vì các tác nhân cài đặt như một dịch vụ Windows, nhưng theo mặc định không tự động khởi động khi hệ thống được khởi động. Bạn cũng có thể cài đặt tác nhân một cách lén lút, sử dụng các công cụ như psexec.exe (<http://technet.microsoft.com/en-us/sysinternals/bb897553.aspx>) hoặc xcmd.exe (<http://feldkir.ch/xcmd.htm>). Tài liệu PDF Triển khai từ xa (Tàng hình) của Phản hồi FEE trên Windows Systems, trực tiếp nằm trong thư mục ch3 trên phương tiện đi kèm với cuốn sách này, minh họa rõ ràng các bước cần thiết để triển khai F-Feedback EE từ xa (hoặc theo cách lén lút) qua mạng. Vào mùa xuân năm 2009, Matt Shannon đã phát hành Bảng điều khiển quản lý doanh nghiệp F-Feedback, hay FEMC. FEMC là một công cụ dựa trên GUI, loại bỏ hoàn toàn tất cả các nỗ



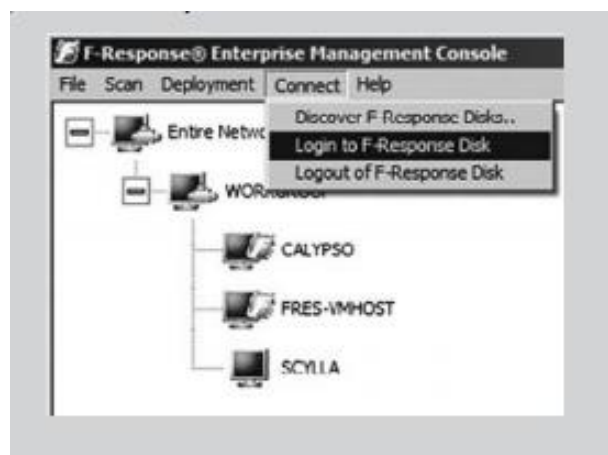
lực cần thiết để triển khai Phiên bản doanh nghiệp theo cách thủ công. Nhấp qua người dùng

giao diện, bộ phản hồi (hoặc nhà tư vấn) có thể định vị các hệ thống trong miền (hoặc nhóm làm việc) để cài đặt F-Feedback EE, như Hình 4 minh họa.



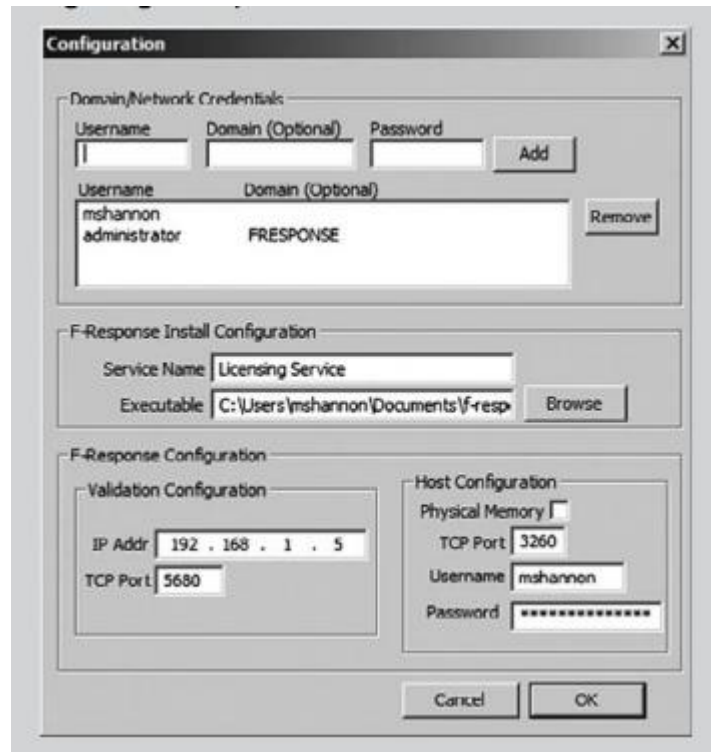
*Hình 4: Giao diện người dùng FEMC*

Khi các hệ thống đã được định vị và chọn, việc triển khai và khởi động dịch vụ Feedback F chỉ cần một vài lần nhấp chuột, như minh họa một phần trong Hình 5. Đây là trường hợp cho dù bạn muốn cài đặt nó trên một hệ thống hoặc trên một tá hệ thống.



*Hình 5: Đăng nhập vào F-Feedback EE thông qua Giao diện người dùng FEMC*

Tất cả các phản hồi cần làm là chọn các hệ thống mà người dùng muốn triển khai Feedback F, và sau đó với một vài lần nhấp chuột sẽ tự động triển khai tác nhân. Ngay cả cấu hình của tệp .ini cũng được xử lý tự động thông qua giao diện người dùng, như Hình 6 minh họa.



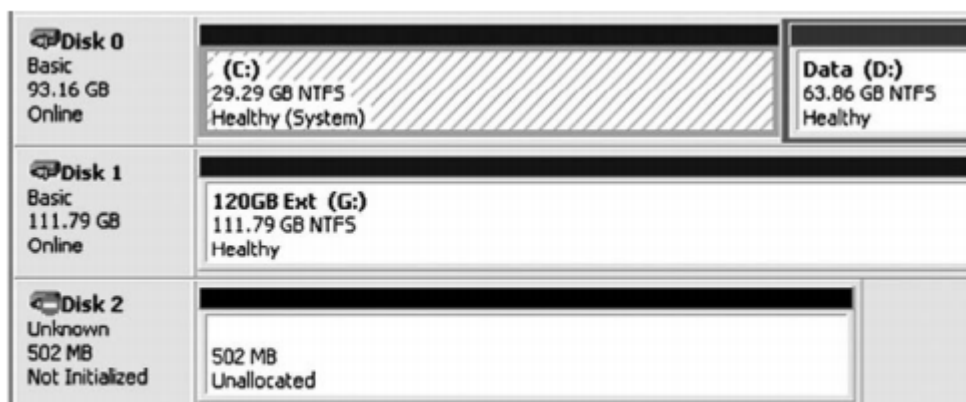
*Hình 6: Cấu hình F-Feedback EE thông qua Giao diện người dùng FEMC*

Thông tin được nhập vào phần Thông tin xác thực tên miền / mạng của hộp thoại Cấu hình được hiển thị trong Hình 6 có sẵn trong suốt thời gian của phiên, nhưng không được lưu hoặc lưu giữ dưới bất kỳ hình thức nào khi đóng FEMC. Làm việc với FEMC, tôi thực sự phải mất nhiều nỗ lực hơn để chơi BrickBreaker trên BlackBerry của mình so với việc triển khai F-Feedback EE trên một số hệ thống, kết nối với từng hệ thống và sau đó gỡ cài đặt hoàn toàn tác nhân. FEMC

đầy chức năng của một công cụ cực kỳ mạnh mẽ và có giá trị về phía trước bằng một bước nhảy lượng tử (không phải lấy đi từ Scott Bakula Thở).

Vì bộ nhớ vật lý (RAM) không chứa bảng phân vùng, hệ thống hô hấp phản hồi sẽ không nhận ra kết nối đến hệ thống từ xa Bộ nhớ vật lý của hệ thống từ xa như một ổ đĩa.

Tuy nhiên, kết nối có thể được nhìn thấy thông qua Bảng điều khiển Microsoft Management Console (MMC), như Hình 7 cho thấy.

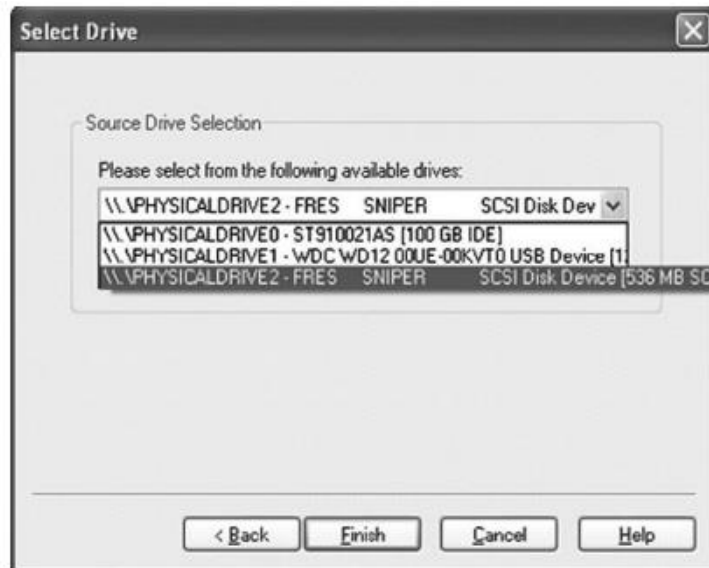


*Hình 7: F-Response EE Connection to Remote RAM Seen via Disk Management MMC*

Khi kết nối với hệ thống từ xa tới RAM đã được xác minh, bạn có thể sử dụng các công cụ như FTK Imager để thực hiện kết xuất bộ nhớ. Hình 8 minh họa việc chọn hệ thống từ xa RAM RAM thông qua FTK Imager.

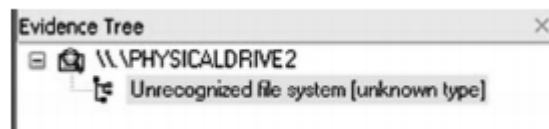
Khi kết nối với hệ thống từ xa, RAM đã được xác minh, bạn có thể sử dụng các công cụ chẳng hạn như FTK Imager để thực hiện kết xuất bộ nhớ. Hình 8 minh họa việc chọn điều khiển từ xa

Hệ thống RAM thông qua FTK Imager.



*Hình 8: Chọn RAM hệ thống từ xa thông qua FTK Imager*

Hình 9 minh họa RAM được chọn từ Hình 8 khi nó xuất hiện trong Cây chứng cứ hình ảnh FTK.



*Hình 9: RAM được chọn trong cây bằng chứng FTK Imager*

Bây giờ có thể lấy RAM từ hệ thống từ xa bằng FTK Imager. Các công cụ khác cũng có thể được sử dụng; Phản hồi F là một khung công cụ không biết công cụ, vì nó không yêu cầu bạn sử dụng một công cụ cụ thể. Người trả lời có thể sử dụng EnCase, X-Way Forensics hoặc thậm chí các phiên bản dd.exe để kết xuất RAM từ hệ thống từ xa. Một lần nữa, kết nối chỉ đọc và như được minh họa trong Hình 8 và 9, RAM RAM hệ thống từ xa xuất hiện trên hệ thống Hồi đáp với tư cách là một đối tượng \. \ PhysDrive (tức là, \. \ PhysDrive2 hoặc \. \ PhysDrive3)

## Tóm tắt

John Sawyer đã đăng lên blog SANS Forensics (<http://sansforensics.wordpress.com/2008/12/13/windows-physical-memory-finding-the-right-tool-for-the-job/>) vào ngày 13 tháng 12, 2008, liệt kê một số công cụ khác nhau có sẵn để thu thập (cũng như phân tích) các bộ nhớ từ các hệ thống Windows, cùng với các mô tả ngắn gọn về từng công cụ. Ngoài bài đăng trên blog của John, các bài đăng khác (lên blog và danh sách thảo luận) đã đưa ra nhận xét và giải quyết mối lo ngại về tính khả dụng và việc sử dụng các công cụ khác nhau để bỏ bộ nhớ từ các hệ thống Windows. Theo nhiều cách, chương này đóng vai trò là bước khởi đầu trong lĩnh vực công cụ này, vì tôi không nghi ngờ rằng giữa thời gian tôi gửi chương này cho nhà xuất bản và thời điểm cuốn sách hoàn thành có sẵn, các công cụ sẽ thay đổi. Chính điều này đã xảy ra giữa thời điểm tôi gửi chương này để xem xét kỹ thuật và khi tôi nhận được biên tập viên công nghệ, nhận xét, trong thời gian tiếp theo, HBGary đã phát hành FastDump Pro và thông báo cho công chúng biết. Tôi tin chắc rằng các công cụ bán phá giá bộ nhớ (và phân tích) sẽ thay đổi, làm như vậy theo thứ tự tương đối ngắn.

Cho đến nay, trong phần này, bạn đã thấy một số công cụ có sẵn để loại bỏ nội dung của bộ nhớ vật lý từ các hệ thống Windows, nhưng ít thôi. Trong một nỗ lực để cung cấp nhiều hơn một so sánh của một số các công cụ này, tôi đã chạy một số thử nghiệm hạn chế của riêng tôi. Sử dụng các dòng lệnh (trong trường hợp Memoryze, tệp memorydd.bat) cho mỗi công cụ đã thảo luận trước đây trong chương này có thể được chạy từ ổ đĩa CD hoặc USB, tôi đã chạy từng cái (ngoại trừ winen.exe) trên hệ thống Dell Latitude D820 với 4 GB RAM được cài đặt, chạy Windows XP Service Pack 3. Quá trình thử nghiệm của tôi chỉ đơn giản là khởi động hệ thống, chạy một trong các công cụ và khi công cụ đã hoàn thành việc kết xuất bộ nhớ, hãy khởi động lại hệ thống để chạy các công cụ tiếp theo.

Khi làm như vậy, tôi đã lưu các tệp bộ nhớ để tôi có thể biết kích thước của chúng liên quan đến các tệp được tạo ra bởi các công cụ khác; kết quả cuối cùng của tôi xuất hiện như sau, được liệt kê theo kích thước theo byte và tên tệp đầu ra:

```
3,210,854,400 win32dd-l0-xp.img
3,210,854,400 win32dd-l1-xp.img
3,210,854,400 mdd-xp-2.img
3,210,854,400 mdd-xp.img
3,211,329,536 dd_xp.img
3,211,329,536 niggilant_xp.img
3,211,333,632 fdpro-xp.bin
3,211,334,904 fdpro-xp.hpak
3,211,329,536 memory.1eld2b07.img (Memoryze)
```

Như bạn có thể thấy, thông tin trên bao gồm tên của công cụ và một số thông tin nhận dạng bổ sung từ dòng lệnh trong tên của tệp kết xuất đầu ra. Ngoài ra, rõ ràng rằng có một số biến thể trong kích thước của các tệp kết quả, có thể là do quá trình mỗi công cụ được sử dụng. Như mong đợi, ngoại lệ cho điều này là tệp kết xuất định dạng *.hpak* do FastDump Pro tạo ra, bộ chuyển đổi trang đã được sử dụng để kết hợp tệp trang trong quy trình kết xuất. Kích thước tệp có thể sẽ lớn hơn đáng kể nếu có hoạt động lớn hơn trên hệ thống và hệ thống được phép chạy lâu hơn.

Tại thời điểm này, rõ ràng là công cụ bạn nên sử dụng để thu thập nội dung của bộ nhớ vật lý thực sự phụ thuộc vào một số yếu tố, bao gồm cả phiên bản Windows (không chỉ Windows XP so với Vista, mà còn 32- so với 64 bit) , dung lượng bộ nhớ vật lý được cài đặt trong hệ thống và các công cụ phân tích của bạn.

### **Chú ý:**

Cũng như việc sử dụng bất kỳ công cụ nào mà bạn đã tải xuống từ Internet, hãy chắc chắn rằng bạn đã thử nghiệm các công cụ đó và bạn đã đọc và bạn hiểu

thỏa thuận cấp phép về việc sử dụng chúng. Một số công cụ, mặc dù cực kỳ mạnh mẽ và hữu ích, không thể được sử dụng bởi các chuyên gia tư vấn. Ngoài ra, bạn cần phải nhận thức được các chức năng của các công cụ khác nhau. Như đã thảo luận trong Chương 1, nhiều công cụ có sẵn từ Microsoft (trước đây là Sysinternals) tạo các mục đăng ký khi chạy. *Winen.exe* thực hiện một cái gì đó tương tự, trong đó nó sẽ ghi trình điều khiển của nó (*winen\_.sys*) vào cùng thư mục mà tệp được chạy và sẽ tạo một khóa con Dịch vụ trong Registry. Mấu chốt ở đây, như với tất cả các hoạt động của Responder, không phải là tránh sử dụng một công cụ vì nó để lại “vết” hoặc các ảnh hưởng khi sử dụng nó, mà là để hiểu thực tế này và kết hợp sự hiểu biết đó vào phương pháp và tài liệu của bạn.

#### *1.4.10. Alternative Approaches for Dumping Physical Memory*

Phần mềm mà chúng tôi đã thảo luận cho đến nay không phải là phương tiện duy nhất mà nội dung của bộ nhớ vật lý có thể được đổ từ hệ thống trực tiếp; một số phương pháp thay thế đã được đưa ra trong quá khứ. Một số phương thức sử dụng chức năng riêng vốn có của hệ điều hành hoặc cho các nền tảng ảo hóa (như VMware). Như chúng ta sẽ thấy, các phương pháp khác để loại bỏ bộ nhớ vật lý từ một hệ thống có cách tiếp cận trực tiếp hơn.

#### *1.4.11. Hardware Devices*

Vào tháng 2 năm 2004, Tạp chí phân tích kỹ thuật số đã xuất bản một bài báo của Brian Carrier và Joe Grand có tiêu đề Quy trình thu thập bộ nhớ dựa trên phân cứng cho phân tích kỹ thuật số. Trong bài báo, Brian và Joe đã trình bày khái niệm về một thẻ mở rộng phân cứng có tên là *Tribble* có thể được sử dụng để truy xuất nội dung của bộ nhớ vật lý vào thiết bị lưu trữ ngoài. Điều này sẽ cho phép một nhà phân tích lấy lại bộ nhớ dễ mất từ hệ thống mà không cần đưa ra bất kỳ mã mới nào hoặc dựa vào mã không đáng tin cậy để thực hiện trích xuất. Trong bài

báo, các tác giả tuyên bố rằng họ đã chế tạo một thiết bị Tribble , được thiết kế như một thẻ mở rộng Công nghiệp Thẻ thanh toán (PCI) có thể sử dụng vào PC bus. Các thiết bị phần cứng khác có sẵn cho phép bạn snapshot của bộ nhớ vật lý và phần lớn nhằm mục đích gỡ lỗi hệ thống phần cứng. Những thiết bị này cũng có thể được sử dụng cho phân tích.

Như được minh họa trong DFRWS 2005 Memory Challenge (đã đề cập trước đó trong chương này), một trong những hạn chế của cách tiếp cận dựa trên phần mềm để lấy bộ nhớ dễ mất là chương trình mà nhà phân tích đang sử dụng phải được nạp vào bộ nhớ. Sau đó, đặc biệt là trên các hệ thống Windows, chương trình có thể (tùy thuộc vào thiết kế của nó) dựa vào mã hoặc thư viện không tin cậy (DLL) mà kẻ tấn công đã phá hoại. Hãy cùng kiểm tra những ưu và nhược điểm của một thiết bị như vậy:

Các Hardware Devices như Tribble không phô trương và dễ dàng truy cập. Việc bỏ nội dung của bộ nhớ vật lý theo cách này sẽ không giới thiệu phần mềm mới hoặc bổ sung nào cho hệ thống, giảm thiểu khả năng dữ liệu bị che khuất theo một cách nào đó. Tuy nhiên, hạn chế chính của việc sử dụng phương pháp dựa trên phần cứng là phần cứng cần phải được cài đặt trước khi xảy ra sự cố. Tại thời điểm này, các thiết bị Tribble không có sẵn rộng rãi. Các thiết bị phần cứng khác có sẵn và dành cho gỡ lỗi phần cứng, nhưng chúng vẫn phải được cài đặt trước khi xảy ra sự cố.

#### *1.4.12. FireWire*

Do đặc thù kỹ thuật của các thiết bị và giao thức FireWire, với phần mềm phù hợp, phân tích viên có thể thu thập nội dung của bộ nhớ vật lý từ hệ thống. Các thiết bị FireWire sử dụng truy cập bộ nhớ trực tiếp (DMA), có nghĩa là chúng có thể truy cập bộ nhớ hệ thống mà không cần phải đi qua CPU. Các thiết bị này có



thể đọc và / hoặc ghi vào bộ nhớ với tốc độ nhanh hơn nhiều so với các hệ thống không sử dụng DMA. Phân tích viên sẽ cần một thiết bị điều khiển có chứa phần mềm thích hợp và có khả năng viết lệnh vào một khu vực cụ thể trong không gian bộ nhớ của thiết bị FireWire. Ánh xạ bộ nhớ được thực hiện trong phần cứng mà không cần thông qua hệ điều hành máy chủ, cho phép truyền dữ liệu có độ trễ thấp tốc độ cao.

Adam Boileau (xem [www.storm.net.nz/projects/16](http://www.storm.net.nz/projects/16)) đã đưa ra một cách để trích xuất bộ nhớ vật lý từ một hệ thống sử dụng Linux và Python. Phần mềm được sử dụng cho phương pháp thu thập này chạy trên Linux và dựa vào hỗ trợ cho thiết bị / dev / raw1394 cũng như thư viện pythonraw1394 của Adam, thư viện libraw1394 và Swig (phần mềm giúp các tệp tiêu đề C / C ++ có thể truy cập được bằng các ngôn ngữ khác bằng cách tạo trình bao bọc mã). Trong các cuộc biểu tình của mình, Adam thậm chí còn bao gồm việc sử dụng một công cụ sẽ thu thập nội dung của RAM từ hệ thống Windows với màn hình bị khóa, phân tích mật khẩu, sau đó Adam đăng nhập vào hệ thống.

Jon Evans, một sĩ quan thuộc sở cảnh sát Gwent ở Anh, đã cài đặt các công cụ của Adam, và thu thập thành công nội dung của bộ nhớ vật lý từ các hệ thống Windows cũng như từ các phiên bản Linux khác nhau. Là một phần trong luận văn thạc sĩ của mình, Jon đã viết một cái nhìn tổng quan về cách cài đặt, thiết lập và sử dụng các công cụ của Adam trên một số nền tảng Linux khác nhau, bao gồm Knoppix v.5.01, Gentoo Linux 2.6.17 và BackTrack, từ Remote-Exploit.org . Khi tất cả các gói cần thiết (bao gồm các công cụ của Adam) đã được tải xuống và cài đặt, Jon sẽ tiến hành quá trình xác định các cổng FireWire và lừa hệ thống Windows mục tiêu vào Nghĩ rằng hệ thống Linux là một iPod bằng cách sử dụng lệnh *romtool* của Linux để tải một tệp dữ liệu chứa Thanh ghi trạng thái điều khiển (CSR) cho iPod (tệp CSR được cung cấp với các công cụ của Adam). Dưới đây là những ưu và nhược điểm của phương pháp này:

Nhiều hệ thống hiện có ngày nay có giao diện FireWire/IEEE 1394 được tích hợp ngay trên bo mạch chủ, tăng khả năng tiếp cận tiềm năng của bộ nhớ vật lý bằng phương pháp này. Tuy nhiên, Arne Vidstrom đã chỉ ra một số vấn đề kỹ thuật (xem <http://ntsecurity.nu/onmymind/2006/2006-09-02.html>) về cách bỏ nội dung của bộ nhớ vật lý qua FireWire có thể dẫn đến treo hoặc trong các phần của bộ nhớ bị bỏ lỡ. George M. Garner, Jr., đã lưu ý trong một trao đổi email trong danh sách gửi thư vào tháng 10 năm 2006 rằng trong thử nghiệm hạn chế, có sự khác biệt đáng chú ý về sự bù đắp quan trọng giữa một tệp RAM được thu thập bằng kỹ thuật FireWire và một được thu thập bằng phần mềm của chính George. Sự khác biệt này chỉ có thể được giải thích là một lỗi trong phương pháp thu thập. Hơn nữa, phương pháp này đã gây ra Màn hình xanh chết chóc (BSOD, được thảo luận thêm trong giấy lát) trên một số hệ thống Windows mục tiêu, có thể là do bản chất của phần cứng FireWire trên hệ thống.

#### *1.4.13. Crash Dumps*

Chúng tôi đã nhìn thấy tất cả các lỗi tại điểm này hay điểm khác; trong hầu hết các trường hợp, chúng biểu hiện như một Màn hình xanh khét tiếng (a.k.a. BSOD). Trong hầu hết các trường hợp, đó là một sự phiền toái, nếu không chỉ ra một vấn đề lớn hơn nhiều. Tuy nhiên, nếu bạn muốn có được một bản sao nguyên sơ, chưa được xử lý của nội dung RAM từ hệ thống Windows, có lẽ cách duy nhất để làm điều đó là tạo ra một bản sửa lỗi hoàn toàn. Điều này là do khi xảy ra sự cố crash, trạng thái hệ thống bị đóng băng và nội dung của RAM (cùng với khoảng 4 Kb thông tin tiêu đề) được ghi vào đĩa. Điều này bảo tồn trạng thái của hệ thống và đảm bảo rằng không có thay đổi nào được thực hiện đối với hệ thống, bắt đầu từ thời điểm đồ vỡ sự cố được bắt đầu.

Thông tin này có thể cực kỳ có giá trị đối với một phân tích viên. Đầu tiên, nội dung của tệp sự cố là snapshot của hệ thống, bị đóng băng trong thời gian. Tôi đã tham gia vào một số cuộc phân tích trong đó các bãi đổ vỡ đã được tìm thấy và sử dụng để xác định nguyên nhân gốc rễ, chẳng hạn như con đường lây nhiễm hoặc thỏa hiệp. Thứ hai, Microsoft cung cấp các công cụ để phân tích các sự cố không chỉ trong Debugging Tools của Microsoft mà còn trong Trình phân tích không gian bộ nhớ Kernel dựa trên Debugging Tools.

Nghe có vẻ là một thỏa thuận tốt, phải không? Rốt cuộc, ngoài việc có tệp 1GB được ghi vào ổ cứng, có thể ghi đè bằng chứng (và không thực sự giảm thiểu tác động của cuộc phân tích của bạn lên hệ thống), không có giới hạn nào đối với phương pháp này, phải không? Trong một số trường hợp, đây là sự thật hoặc bạn có thể chấp nhận điều kiện đó, tùy thuộc vào hoàn cảnh. Tuy nhiên, vẫn còn một vài vấp ngã. Đầu tiên, không phải tất cả các hệ thống tạo ra các bãi đổ vỡ đầy đủ theo mặc định. Thứ hai, theo mặc định, các hệ thống Windows không tạo ra các sự cố theo lệnh.

Vấn đề đầu tiên tương đối đơn giản để giải quyết, theo bài viết của Microsoft Knowledge Base Q254649 (<http://support.microsoft.com/kb/254649>). Bài viết này liệt kê ba loại bãi lỗi: nhỏ (64 KB), hạt nhân và các lỗi hoàn toàn. Chúng tôi đang tìm kiếm tệp sự cố hoàn toàn vì nó chứa nội dung hoàn chỉnh của RAM. Bài báo cũng nói rằng Windows 2000 Pro và Windows XP (cả Pro và Home) sẽ tạo ra các bản sửa lỗi nhỏ và Windows 2003 (tất cả các phiên bản) sẽ tạo ra các bản sửa lỗi hoàn toàn. Kinh nghiệm của tôi với Windows Vista RC1 là theo mặc định, nó sẽ tạo ra các bãi đổ vỡ nhỏ.

#### **Ghi chú:**

Bài viết Microsoft Knowledge Base 235496 ( <a href="http://support.microsoft.com/kb/235496">http://support.microsoft.com/kb/235496</a> ) chỉ định các mục đăng ký có chứa
---

thông tin cấu hình cho các hệ thống Windows đối với tệp memory.dmp là kết quả của kết xuất lỗi. Vị trí mặc định cho tệp memory.dmp là %SystemRoot%\memory.dmp, nhưng quản trị viên có thể chỉ định một vị trí khác thông qua giá trị DumpFile Registry.

Đồng thời, bài viết Microsoft Knowledge Base Q274598 (<http://support.microsoft.com/kb/274598>) nói rằng các bãi đổ vỡ hoàn toàn không có sẵn trên các hệ thống có hơn 2 GB RAM. Theo bài báo, điều này phần lớn là do các yêu cầu về không gian (nghĩa là đối với các hệ thống đã kích hoạt hoàn toàn sự cố, tệp trang phải lớn bằng nội dung của RAM + 1 MB) cũng như thời gian cần thiết để hoàn thành quá trình đổ vỡ.

Bài viết Microsoft Knowledge Base Q307973 mô tả cách đặt toàn bộ phạm vi của các tùy chọn khôi phục và lỗi hệ thống. Các cài đặt này dành cho quản trị viên hệ thống và người quản lý công nghệ thông tin (CNTT) đang thiết lập và định cấu hình hệ thống trước khi xảy ra sự cố, nhưng cài đặt khóa Sổ đăng ký có thể cung cấp một số manh mối quan trọng cho một nhà phân tích. Ví dụ: nếu hệ thống được cấu hình (theo mặc định hoặc theo cách khác) để tạo kết xuất sự cố hoàn toàn và quản trị viên báo cáo đã nhìn thấy BSoD, phân tích viên sẽ mong đợi thấy tệp kết xuất sự cố hoàn toàn trên hệ thống.

### **Chú ý:**

Các nhà phân tích phải cực kỳ cẩn thận khi làm việc với các tệp kết xuất sự cố, đặc biệt là từ các hệ thống xử lý nhưng không nhất thiết phải lưu trữ dữ liệu nhạy cảm. Trong một số trường hợp, các bãi đổ vỡ đã xảy ra trên các hệ thống xử lý thông tin như sổ thẻ tín dụng, sổ An sinh xã hội hoặc tương tự, và các bãi đổ vỡ này đã được tìm thấy có chứa thông tin nhạy cảm đó. Mặc dù các lập trình viên đặc biệt đã viết ứng dụng để không có thông tin cá nhân nhạy cảm nào được

lưu cục bộ trên hệ thống, một tệp sự cố đã ghi nội dung của bộ nhớ vào ổ cứng.

Vì vậy, hãy nói với các tùy chọn cấu hình khôi phục và khôi phục hệ thống trên một hệ thống được đặt trước thời hạn (như một phần của chính sách cấu hình cho các hệ thống) để thực hiện xử lý sự cố hoàn toàn. Làm thế nào để phân tích viên khuyến khích một hệ thống thực hiện lệnh đổ vỡ trên lệnh, khi nó cần? Hóa ra là có một khóa Registry (xem bài viết Knowledge Base Q244139, có sẵn tại <http://support.microsoft.com/kb/244139>) có thể được đặt để gây ra lỗi đổ vỡ khi phím Ctrl bên phải được giữ và phím Phím Scroll Lock được nhấn hai lần. Tuy nhiên, khi khóa này được đặt, hệ thống phải được khởi động lại để cài đặt có hiệu lực. Hãy cùng xem xét những ưu và nhược điểm của kỹ thuật này:

Kết xuất bộ nhớ thông qua kết xuất sự cố có lẽ là phương pháp chính xác duy nhất về mặt kỹ thuật (mặc dù không phải là âm thanh forensally âm thanh) để tạo hình ảnh về nội dung của RAM. Điều này là do khi hàm API KeBugCheck được gọi, toàn bộ hệ thống bị tạm dừng và nội dung của RAM được ghi vào tệp trang, sau đó chúng được ghi vào một tệp trên ổ cứng hệ thống (ghi đè dữ liệu). Hơn nữa, Microsoft cung cấp các Debugging Toolscũng như Trình phân tích không gian bộ nhớ Kernel (bao gồm một công cụ, trình cắm và giao diện người dùng) để phân tích các tệp kết xuất sự cố. Một số hệ thống Windows không tạo ra các sự cố hoàn toàn theo mặc định (ví dụ: Vista RC1; Tôi gặp sự cố với trình điều khiển khi tôi cài đặt Vista RC1 lần đầu tiên và tôi sẽ nhận được BSoD bất cứ khi nào tôi cố gắng tắt máy, dẫn đến các tệp kết xuất nhỏ ).

Ngoài ra, việc sửa đổi một hệ thống để chấp nhận chuỗi phím tắt để tạo kết xuất sự cố đòi hỏi phải khởi động lại và phải được thực hiện trước thời hạn để được sử dụng hiệu quả cho phản ứng sự cố. Ngay cả khi thay đổi cấu hình này đã được thực hiện, quá trình kết xuất sự cố vẫn sẽ tạo một tệp có kích thước tương

đương với bộ nhớ vật lý trên ổ cứng. Để làm như vậy, như đã nêu trong bài viết Knowledge Base Q274598, tệp trang phải được định cấu hình bằng với ít nhất kích thước của bộ nhớ vật lý cộng với 1 MB. Đây là một bước bổ sung phải được sửa chữa để sử dụng phương pháp chụp nội dung của bộ nhớ vật lý này; Nó một trong những điều không thường xuyên theo sau.

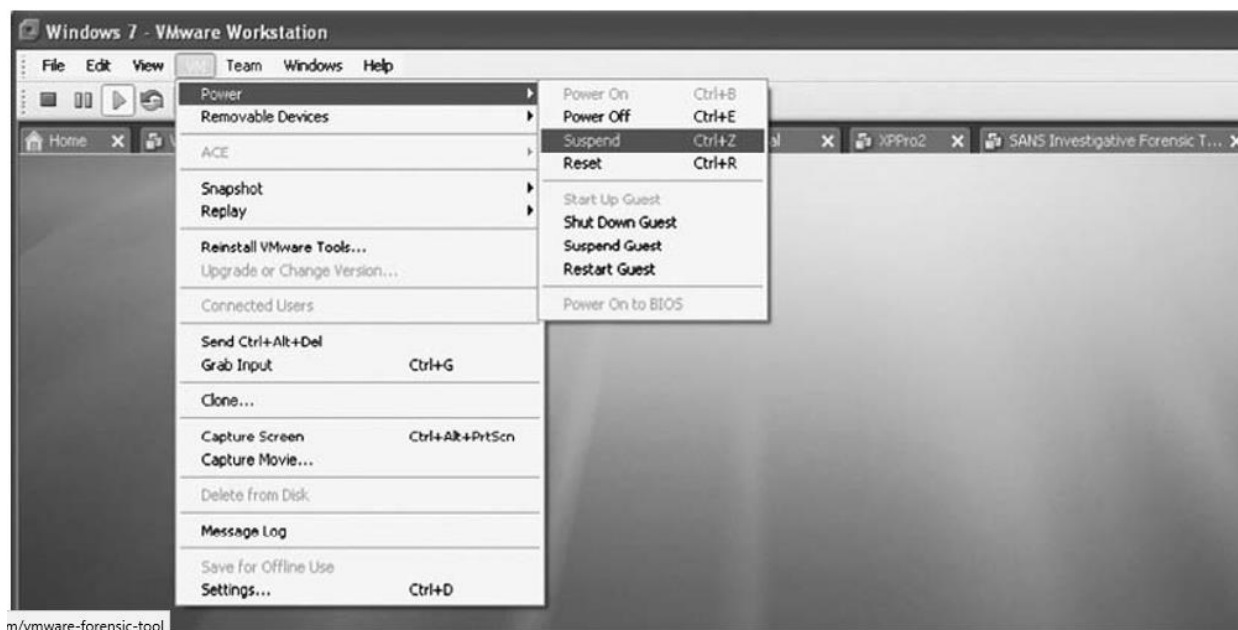
**Chú ý:**

Một bài viết hỗ trợ có sẵn tại Citrix Web (<http://support.citrix.com/article/CTX107717&parentCategoryID=617>) cung cấp phương pháp sử dụng livekd.exe (<http://technet.microsoft.com/en-us/sysinternals/bb897415.aspx>) và Debugging Tools của Microsoft để tạo ra một kết xuất hạt nhân đầy đủ của bộ nhớ vật lý. Khi livekd.exe được khởi chạy, lệnh .dump / f <filepath> được sử dụng để tạo tệp kết xuất. Bài viết hỗ trợ bao gồm cảnh báo rằng các kết xuất RAM được tạo theo cách này có thể không nhất quán vì kết xuất có thể mất một lượng thời gian đáng kể và hệ thống đang hoạt động và tiếp tục chạy trong kết xuất bộ nhớ.

#### *1.4.14. Virtualization*

VMware là một sản phẩm ảo hóa phổ biến (VMware Workstation 5.5.2 được sử dụng rộng rãi trong cuốn sách này), vì một điều, cho phép tạo ra các mạng giả lập sử dụng phần cứng của một hệ thống. Sử dụng mạng giả lập này có nhiều lợi ích. Ví dụ: bạn có thể thiết lập một hệ điều hành client và tạo một snapshot của hệ thống đó một khi bạn đã cấu hình nó theo nhu cầu của bạn. Từ đó, bạn có thể thực hiện tất cả các cách kiểm tra, bao gồm cài đặt và giám sát phần mềm độc hại ; bạn sẽ luôn có thể reverse lại snapshot và bắt đầu lại. Tôi thậm chí đã thấy các hệ thống sản xuất hoạt động chạy từ các phiên VMware.

Khi bạn đang chạy một phiên VMware, bạn có thể tạm dừng phiên đó, tạm thời đóng băng nó. Hình 10 minh họa các mục menu để tạm dừng phiên VMware.



Hình 10: Các mục menu để tạm dừng một phiên trong VMware Workstation 6.5

Khi phiên VMware bị treo, nội dung của bộ nhớ vật lý được chứa trong một tệp có phần mở rộng *.vmem*. Định dạng của tệp này rất giống với định dạng của bộ nhớ thô, kiểu dd và trên thực tế, nó có thể được phân tích cú pháp và phân tích với cùng các công cụ đã thảo luận trước đó trong chương.

VMware là sản phẩm ảo hóa duy nhất hiện có. Những người khác bao gồm VirtualPC từ Microsoft, cũng như Bochs phần mềm miễn phí (<http://bochs.sourceforge.net/>). Không có sản phẩm ảo hóa nào trong số này đã được thử nghiệm để xác định xem chúng có thể tạo ra các bộ nhớ vật lý hay không; tuy nhiên, nếu đây là một tùy chọn có sẵn cho bạn, việc tạm dừng phiên VMware để có được bộ nhớ vật lý nhanh chóng và dễ dàng và giảm thiểu sự tương tác của bạn với và tác động đến hệ thống.

### Chú ý:

Vào tháng 5 năm 2006, Brett Shavers đã viết một bài viết tuyệt vời cho trang ForensicFocus Web, có tựa đề là “VMware as a forensic tool “ (có sẵn tại [www.forensicfocus.com/vmware-forensic-tool](http://www.forensicfocus.com/vmware-forensic-tool)).

#### 1.4.15. Hibernation File

Khi hệ thống Windows (Windows 2000 trở lên) “hibernates”, thì Trình quản lý nguồn sẽ lưu nội dung được nén của bộ nhớ vật lý vào một tệp có tên *hiberfil.sys* trong thư mục gốc của ổ đĩa hệ thống. Tệp này đủ lớn để chứa các nội dung không nén của bộ nhớ vật lý, nhưng nén được sử dụng để giảm thiểu I / O của đĩa và để cải thiện hiệu suất đóng băng. Trong quá trình khởi động, nếu tệp *hiberfil.sys* hợp lệ được đặt, Trình tải NT (NTLDR) sẽ tải nội dung của tệp vào bộ nhớ vật lý và chuyển điều khiển sang mã trong nhân xử lý hoạt động lại hệ thống sau khi đóng băng (tải trình điều khiển, v.v. ). Chức năng này thường được tìm thấy trên các hệ thống máy tính xách tay. Dưới đây là những ưu và nhược điểm:

Phân tích nội dung của tệp đóng băng có thể cho bạn manh mối về những gì đang xảy ra trên hệ thống tại một số điểm trong quá khứ. Matthieu Suiche đã giải mã định dạng tệp đóng băng và trình bày kết quả của mình tại hội nghị BlackHat USA 2008 . Bài thuyết trình này đã tiếp nối bài viết trước đó của anh ấy về chủ đề này từ tháng 2 năm 2008 có tựa đề là “Sandman Project” và bài viết trước đó của anh ấy từ năm 2007 với Nicolas Ruff. Công cụ Sandman có sẵn từ [www.msuiche.net/carget/sandman/](http://www.msuiche.net/carget/sandman/) và chức năng được tích hợp vào Volatility Framework (sẽ thảo luận sau trong chương này).

Một điều khác cần xem xét về các tệp đóng băng là chức năng này có thể là tùy chọn duy nhất có sẵn để chụp toàn bộ nội dung của bộ nhớ vật lý. Một số công cụ có sẵn để thu thập nội dung của bộ nhớ vật lý có thể có các vấn đề về bộ phận dữ liệu (cụ thể hơn, chúng có thể khiến các hệ thống có bộ nhớ vật lý hơn 4 GB, bao gồm cả hệ thống 64 bit, bị lỗi) hoặc có thể không khả dụng khả thi để sử dụng trong một số môi trường. Sử dụng *powercfg.exe* để bật chế độ đóng băng (nếu chưa được bật) và sau đó sử dụng một số cơ chế hoặc công cụ khác để buộc hệ thống



đóng băng có thể là lựa chọn duy nhất để có được kết xuất bộ nhớ. Bạn có thể buộc một hệ thống với chế độ đóng băng được kích hoạt để đóng băng bằng cách sử dụng tiện ích *pssshutdown.exe* của Microsoft (<http://technet.microsoft.com/en-us/sysinternals/bb897541.aspx>) với tùy chọn *lång* hoặc bằng cách tạo một tệp bó chứa dòng:

***%windir%\System32\rundll32.exe powrprof.dll,SetSuspendState Hibernate***

Tệp hibernation được nén và trong hầu hết các trường hợp sẽ không chứa nội dung hiện tại của bộ nhớ. Nhờ vào công việc được thực hiện bởi Matthieu và Nicolas, tập tin đóng băng có thể được truy cập theo cách tương tự như một tệp bộ nhớ trực tiếp, vì vậy tôi có thể thực sự nghĩ về bất kỳ điều gì khi sử dụng nó. Ngoài việc kết xuất bộ nhớ từ hệ thống trực tiếp, việc có sẵn tệp đóng băng sẽ cung cấp cho bạn nội dung bộ nhớ từ thời điểm trước đó để so sánh kết xuất bộ nhớ của bạn.

#### *1.4.16. Analyzing a Physical Memory Dump*

Với DFRWS 2005 Memory Challenge làm chất xúc tác, các bước đã được thực hiện trong nỗ lực thêm ngữ cảnh vào thông tin tìm thấy trong RAM. Bằng cách định vị các quy trình cụ thể (hoặc các đối tượng khác được duy trì trong bộ nhớ) và các trang bộ nhớ được sử dụng bởi các quy trình đó, các nhà phân tích có thể hiểu rõ hơn về thông tin họ khám phá cũng như thực hiện giám dữ liệu đáng kể bằng cách lọc ra các quy trình và dữ liệu tốt được biết đến trên dữ liệu có vẻ bất thường. Một số cá nhân có các công cụ bằng văn bản có thể được sử dụng để phân tích cú pháp RAM và lấy thông tin chi tiết về các quy trình và các cấu trúc khác.

Vào năm 2007, Aaron Walters đã phát hành Volatility Framework (<https://www.volatilesystems.com/>), một Framework dựa trên Python để phân tích cú pháp bộ nhớ từ (tại thời điểm viết bài này) các hệ thống Windows XP. Vào mùa hè 2008, Aaron đã tổ chức Hội thảo mở về Phân tích (ngay trước hội nghị DFRWS

2008, trong đó các nhà nghiên cứu, nhà phân tích và học viên có thể thảo luận các vấn đề xung quanh thu thập và phân tích bộ nhớ. Phiên bản 1.3 của Volatility Framework đã có sẵn tại hội nghị và thậm chí còn được sử dụng trong cuộc tập trận Forensic Rodeo của DFRWS.

Trong suốt phần còn lại của chương này, chúng tôi sẽ xem xét các công cụ này để thực hiện phân tích các tệp bộ nhớ. Ban đầu, chúng tôi sẽ sử dụng các kết xuất bộ nhớ từ Thử thách bộ nhớ DFRWS 2005 làm ví dụ, ví dụ và trình diễn các công cụ và kỹ thuật để phân tích cú pháp bộ nhớ Windows 2000. Bạn có thể tự hỏi mình, tại sao thậm chí bận tâm với điều đó? Windows 2000 là MS-DOS mới, phải không? Chà, điều đó có lẽ không xa sự thật, nhưng các bãi rác cung cấp một cơ sở tuyệt vời cho các ví dụ vì chúng đã được kiểm tra rất chi tiết. Ngoài ra, họ còn có sẵn miễn phí để tải về và kiểm tra.

#### *1.4.17. Determining*

Bạn đã bao giờ được trao một hình ảnh của một hệ thống, và khi bạn hỏi hệ điều hành là gì /, bạn chỉ đơn giản là đã nhận được Window? Shakespeare hiện không cắt nó ở đây, các bạn ạ, bởi vì một bông hồng của bất kỳ tên nào khác có thể không có mùi ngọt ngào. Khi bạn làm việc với một hình ảnh của một hệ thống, phiên bản Windows của bạn đã đối mặt với các vấn đề, và tùy thuộc vào vấn đề mà bạn đã xử lý, nó có thể rất quan trọng. Điều tương tự cũng đúng khi bạn xử lý tệp kết xuất RAM; trong thực tế, nó có thể thậm chí nhiều hơn như vậy. Như I Minhve đã nêu, các cấu trúc được sử dụng để xác định các luồng và tiến trình trong bộ nhớ không chỉ khác nhau giữa các phiên bản chính của hệ điều hành mà còn trong cùng một phiên bản với các gói dịch vụ khác nhau được cài đặt.

Vì vậy, khi ai đó trao cho bạn một file RAM dump và nói rằng nó là của Windows, bạn có muốn biết làm thế nào để tìm ra điều đó? Rốt cuộc, bạn không

muốn lãng phí nhiều thời gian để chạy tệp kết xuất thông qua mọi công cụ đã biết cho đến khi một trong số chúng bắt đầu tạo ra các lượt truy cập hợp lệ trên các quy trình, phải không? Thông qua thư từ cá nhân (đó là một thuật ngữ ưa thích cho Email e-mail) cách đây một thời gian, Andreas Schuster đã gợi ý cho tôi rằng Windows kernel có thể có thể được tải vào cùng một vị trí trong bộ nhớ mỗi khi Windows khởi động. Bây giờ, vị trí đó có khả năng và sẽ thay đổi cho mọi phiên bản Windows, nhưng cho đến nay nó dường như phù hợp với từng phiên bản. Cách dễ nhất để tìm vị trí này là chạy *LiveKD* như chúng ta đã làm trước đó trong chương này, nhưng đặc biệt lưu ý thông tin mà hiển thị khi nó khởi động (hiển thị ở đây trên hệ thống Windows XP SP2):

**Windows XP Kernel Version 2600 (Service Pack 2) MP (2 procs) Free x86 compatible Product: WinNt,**

**suite: TerminalServer**

**SingleUserTS**

**Built**

**by:2600.xpsp.050329-1536**

**Kernel base = 0x804d7000 PsLoadedModuleList = 0x8055c700**

Chúng tôi quan tâm nhất đến thông tin mà tôi đã in đậm tên địa chỉ của kernel. Chúng tôi lấy *0x80000000* từ địa chỉ đó và sau đó đi đến vị trí kết quả thực tế trong tệp kết xuất. Nếu hai byte đầu tiên được đặt tại địa chỉ đó là MZ, chúng ta có thể có tệp thực thi di động (PE) Windows đầy đủ tại vị trí đó và chúng ta có thể có kernel. Từ thời điểm này, chúng ta có thể sử dụng mã tương tự như những gì trong *Lspi.pl* để phân tách tiêu đề PE và định vị các phần khác nhau trong tệp PE. Vì nhân Windows là một tệp ứng dụng hợp pháp của Microsoft, chúng tôi có thể chắc chắn rằng có một phần tài nguyên trong tệp có chứa phần *VS\_VERSION\_INFO*. Theo thông tin do Microsoft cung cấp liên quan đến các câu

trúc khác nhau tạo nên phần này, sau đó chúng tôi có thể phân tích thông qua nó để tìm chuỗi mô tả tệp.

Trên DVD đi kèm, bạn sẽ tìm thấy một tệp có tên *osid.pl* thực hiện điều đó. *Osid.pl* bắt đầu như *kern.pl* và tìm đường vào tiện ích Rick McQuown's PTFinderFE. Một ngày nọ, Rick hỏi tôi qua e-mail liệu có cách nào để rút ngắn và làm rõ đầu ra không, vì vậy tôi đã thực hiện một số sửa đổi cho tệp (thay đổi đầu ra, thêm một số công tắc, v.v.) và đổi tên nó.

Ở dạng đơn giản nhất, bạn có thể chạy *osid.pl* từ dòng lệnh, chuyển đường dẫn đến tệp hình ảnh dưới dạng đối số duy nhất:

```
C:\Perl\memory>osid.pl d:\hacking\xp-laptop1.img
```

Ngoài ra, bạn có thể chỉ định một tệp cụ thể bằng cách sử dụng khóa -f:

```
C:\Perl\memory>osid.pl -f d:\hacking\xp-laptop1.img
```

Cả hai lệnh này sẽ cung cấp cho bạn cùng một đầu ra; trong trường hợp này, kết xuất RAM được thu thập từ hệ thống Windows XP SP2, do đó tập lệnh trả về XPSP2. Nếu đây là thông tin khá đầy đủ và bạn muốn xem thêm, bạn có thể thêm khóa chuyển đổi (đối với verbose) và tập lệnh sẽ trả về phần sau cho tệp xp-laptop1.img:

```
OS :XPSP2
```

```
Product :Microsoft« Windows« Operating System ver 5.1.2600.2622
```

Như bạn có thể thấy, các chuỗi trong cấu trúc VS\_VERSION\_INFO liên quan đến tên sản phẩm và phiên bản sản phẩm được nối để tạo đầu ra bổ sung. Nếu chúng tôi chạy tập lệnh với cả Bộ chuyển đổi và Bộ chuyển đổi dựa trên tệp kết xuất RAM đầu tiên từ DFRWS Memory Challenge 2005, chúng tôi sẽ nhận được:

```
OS :2000
```

```
Product :Microsoft(R) Windows (R) 2000 Operating System ver 5.00.2195.1620
```

Chạy kịch bản này chống lại các tệp bộ nhớ khác được đề cập trước đó trong chương này minh họa mức độ hoạt động của nó trên các phiên bản Windows khác nhau. Ví dụ: khi chạy với kết xuất bộ nhớ được sử dụng trong ví dụ Memoryze, chúng ta thấy như sau:

```
C      :|Perl|memory>osid.pl      -f      d:|hacking|boomer-win2003.img      -v
OS      :2003
```

***Product :Microsoft« Windows« Operating System ver 5.2.3790.0***

Chạy tập lệnh đối với kết xuất bộ nhớ Windows 2003 khác cho chúng ta kết quả hơi khác nhau:

```
C      :|Perl|memory>osid.pl      -f      d:|hacking|win2k3sp1_physmem.img      -v
OS      :2003SP1
```

***Product :Microsoft« Windows« Operating System ver 5.2.3790.1830***

Kịch bản này cũng hoạt động tốt như nhau đối với các tệp .vmem của VMware. Tôi đã chạy tập lệnh chống lại tệp .vmem từ phiên VMware Windows 2000 và nhận được kết quả đầu ra sau:

```
OS      :2000
```

***Product :Microsoft(R) Windows (R) 2000 Operating System ver 5.00.2195.7071***

Tôi nghĩ rằng hơn bất cứ điều gì khác, điều này chứng tỏ sự tiện ích của các tập lệnh hoặc công cụ như thế này, vì nó cung cấp cho nhà phân tích khả năng ghi lại hoàn toàn các mục khác nhau để phân tích, đặc biệt khi những điều đó có thể không được ghi lại hoàn toàn trong các hoạt động đáp ứng khi dữ liệu ban đầu được thu thập.

### ***Chú ý:***

Hầu hết các công cụ phân tích, được thảo luận sau trong chương này, có khả năng thực hiện chức năng tương tự như vừa được thảo luận. Ví dụ: bạn có thể sử dụng lệnh Nhận dạng biến động để xác định hệ điều hành của kết xuất bộ

nhớ.

#### *1.4.18. Process Basics*

Trong suốt chương này, chúng tôi sẽ tập trung chủ yếu vào việc phân tích thông tin liên quan đến các quá trình từ kết xuất bộ nhớ. Điều này một phần là do thực tế là phần lớn các nghiên cứu và công cụ có sẵn công khai tập trung vào các quy trình như một nguồn thông tin pháp y. Điều đó không có nghĩa là các đối tượng khác trong bộ nhớ nên được loại trừ, mà là hầu hết các nhà nghiên cứu dường như đang tập trung vào các quy trình. Chúng tôi sẽ thảo luận về một phương tiện khác để lấy thông tin từ kết xuất RAM sau chương này, nhưng bây giờ, chúng tôi sẽ tập trung nỗ lực vào các quy trình. Để đạt được điều đó, chúng ta cần có một ý tưởng khá hay về một quy trình mà trông giống như thế nào trong bộ nhớ. Phần sau đây tập trung vào các quy trình trong bộ nhớ Windows 2000, nhưng hầu hết các khái niệm vẫn giống nhau cho tất cả các phiên bản Windows. Sự khác biệt lớn nhất là ở cấu trúc thực tế của chính quá trình và đi sâu vào chi tiết về cấu trúc quy trình trên tất cả các phiên bản Windows nằm ngoài phạm vi của cuốn sách này.

#### *1.4.19. EProcess Structure*

Mỗi quy trình trên hệ thống Windows được thể hiện dưới dạng một quy trình điều hành hoặc khối EProcess. Khối EProcess này là một cấu trúc dữ liệu trong đó các thuộc tính khác nhau của quy trình, cũng như con trỏ tới một số thuộc tính và cấu trúc dữ liệu khác (luồng, khối môi trường quy trình) liên quan đến quy trình, được duy trì. Bởi vì cấu trúc dữ liệu là một chuỗi các byte, với mỗi chuỗi có một ý nghĩa và mục đích cụ thể, các cấu trúc này có thể được đọc và phân tích bởi một nhà phân tích. Tuy nhiên, một điều cần lưu ý là điều duy nhất nhất quán giữa các phiên bản của hệ điều hành Windows liên quan đến các cấu trúc này là chúng không nhất quán. Bạn đã nghe đúng: Kích thước và thậm chí các giá trị của cấu

trúc không chỉ thay đổi giữa các phiên bản hệ điều hành (ví dụ: Windows 2000 sang XP) mà còn giữa các gói dịch vụ của cùng một phiên bản hệ điều hành (Windows XP đến XP SP2).

Andreas Schuster đã thực hiện một công việc tuyệt vời là ghi lại các cấu trúc khối EProcess trong blog của mình. Tuy nhiên, tương đối dễ dàng để xem nội dung của cấu trúc EProcess (hoặc bất kỳ cấu trúc nào khác có sẵn trên Windows). Đầu tiên, tải xuống và cài đặt Debugging Tools của Microsoft và các ký hiệu chính xác cho hệ điều hành và Gói dịch vụ của bạn. Sau đó tải xuống *livekd.exe* từ Sysinternals.com (khi bạn nhập **sysinternals.com** vào thanh địa chỉ của trình duyệt, bạn sẽ tự động được chuyển hướng đến trang web của Microsoft, bởi vì bây giờ, Mark Russinovich đã sử dụng Microsoft từ lâu) và để thuận tiện, sao chép nó vào cùng thư mục với Công cụ gỡ lỗi. Khi bạn đã thực hiện việc này, hãy mở một dấu nhắc lệnh, thay đổi thư mục nơi bạn đã cài đặt Debugging Tools và nhập lệnh sau:

**D:\debug>livekd -w**

Lệnh này sẽ mở WinDbg, giao diện GUI cho các công cụ gỡ lỗi. Để xem toàn bộ nội dung của khối EProcess, trông như thế nào (giống với tất cả các cấu trúc con tạo nên cấu trúc EProcess), hãy nhập `dt -a -b -v _EPROCESS` vào cửa sổ lệnh và nhấn Enter. Cờ -a hiển thị từng thành phần mảng trên một dòng mới, với chỉ mục của nó và cờ -b hiển thị các khối theo cách đệ quy. Cờ -v tạo ra nhiều đầu ra dài dòng hơn, cho bạn biết kích thước tổng thể của từng cấu trúc. Trong một số trường hợp, có thể hữu ích khi bao gồm cờ -r cho đầu ra đệ quy. Phần sau đây minh họa một đoạn trích ngắn từ kết quả của lệnh này, chạy trên hệ thống Windows 2000:

```
kd> dt -a -b -v _EPROCESS
struct _EPROCESS, 94 elements, 0x290 bytes
    +0x000 Pcb : struct _KPROCESS, 26 elements, 0x6c bytes
    +0x000 Header : struct _DISPATCHER_HEADER, 6 elements, 0x10
bytes
```

```

+0x000 Type : Uchar
+0x001 Absolute : Uchar
+0x002 Size : UChar
+0x003 Inserted : UChar
+0x004 SignalState : Int4B
+0x008 WaitListHead : struct _LIST_ENTRY, 2 elements, 0x8 bytes
    +0x000 Flink : Ptr32 to
    +0x004 Blink : Ptr32 to
+0x010 ProfileListHead : struct _LIST_ENTRY, 2 elements, 0x8
bytes
    +0x000 Flink : Ptr32 to
    +0x004 Blink : Ptr32 to
+0x018 DirectoryTableBase : (2 elements)Uint4B

```

Toàn bộ đầu ra dài hơn nhiều (theo tiêu đề, toàn bộ cấu trúc dài 0x290 byte), nhưng đừng lo, chúng tôi sẽ giải quyết các yếu tố quan trọng (từ khía cạnh pháp y / phân tích) của cấu trúc khi chúng tôi tiến hành trong chương này.

### Chú ý:

Windows kernel theo dõi các quy trình hoạt động bằng danh sách liên kết đôi; điều này có nghĩa là mỗi quy trình mà điểm của điểm đến với cả hai quá trình sau nó và quá trình trước nó, trong một danh sách vòng tròn. Hệ điều hành liệt kê một danh sách các quy trình hoạt động bằng cách đi bộ PsActiveProcessList và phát triển danh sách các quy trình hoạt động đã biết. Trong cấu trúc EProcess là một mục LIST\_ENTRY có tên ActiveProcessLinks. Mục này có hai giá trị, nhấp nháy và nhấp nháy, tương ứng là các con trỏ tới các quá trình tiếp theo và trước đó. Nhiều công cụ phân tích bộ nhớ sẽ làm điều tương tự (ví dụ: đi bộ danh sách các quy trình hoạt động) trong tệp kết xuất bộ nhớ, trong khi các công cụ khác sẽ thực hiện phép liệt kê các đối tượng xử lý (ví dụ: lsproc.pl và Biến động), liệt kê các quy trình thậm chí đã thoát. Điều này rất quan trọng, vì một số rootkit (xem Chương 7) ẩn các quy trình bằng cách hủy liên kết các quy trình của chúng khỏi danh sách liên kết đôi này.



Một yếu tố quan trọng của một quy trình mà cấu trúc EProcess chỉ ra là khối môi trường quy trình, hoặc PEB. Cấu trúc này chứa rất nhiều thông tin, nhưng các yếu tố quan trọng đối với chúng tôi, như các nhà phân tích pháp y, là:

Một con trỏ tới cấu trúc dữ liệu của trình tải (được gọi là cấu trúc PPEB\_LDR\_DATA) bao gồm các con trỏ hoặc tham chiếu đến các mô-đun (DLL) được sử dụng bởi quy trình

Một con trỏ đến địa chỉ cơ sở hình ảnh, nơi chúng ta có thể mong đợi để tìm phần đầu của tệp hình ảnh thực thi

Một con trỏ tới cấu trúc tham số tiến trình, chính nó duy trì đường dẫn DLL, đường dẫn đến hình ảnh thực thi và dòng lệnh được sử dụng để khởi chạy tiến trình

Trích xuất thông tin này từ một tệp kết xuất có thể chứng minh là cực kỳ hữu ích cho một nhà phân tích, như bạn sẽ thấy trong suốt phần còn lại của chương này.

#### *1.4.20. Process Creation Mechanism*

Bây giờ bạn đã biết một chút về các cấu trúc khác nhau liên quan đến các quy trình, sẽ rất hữu ích khi biết điều gì đó về cách hệ điều hành sử dụng các cấu trúc đó, đặc biệt là khi tạo ra một quy trình thực tế.

Một số bước được theo sau khi một quy trình được tạo. Các bước này có thể được chia thành sáu giai đoạn (được lấy từ Windows Internals, 4th Edition, Chương 6, bởi Russinovich và Solomon):

Tệp image (.exe) sẽ được thực hiện được mở. Trong giai đoạn này, hệ thống con thích hợp (POSIX, MS-DOS, Win 16, v.v.) được xác định. Ngoài ra, khóa Image File Execution Options Registry (xem Chương 4) được kiểm tra để xem liệu có giá trị Trình gỡ lỗi hay không và nếu có, quá trình bắt đầu lại.

Đối tượng EProcess được tạo. Khối quy trình nhân (KProcess), khối môi trường quy trình và không gian địa chỉ ban đầu cũng được thiết lập.

Các chủ đề ban đầu được tạo ra

Hệ thống con Windows được thông báo về việc tạo quy trình và luồng mới, cùng với ID của trình tạo quy trình và một cờ để xác định xem quy trình có thuộc về quy trình Windows hay không.

Thực hiện các chủ đề ban đầu bắt đầu. Tại thời điểm này, môi trường quy trình đã được thiết lập và các tài nguyên đã được phân bổ cho (các) luồng xử lý để sử dụng.

Việc khởi tạo không gian địa chỉ được hoàn thành, trong bối cảnh của tiến trình và luồng mới

Tại thời điểm này, quy trình hiện tiêu tốn không gian trong bộ nhớ theo cấu trúc EProcess (bao gồm cấu trúc KProcess) và cấu trúc PEB. Quá trình có ít nhất một luồng và có thể bắt đầu tiêu thụ thêm tài nguyên bộ nhớ khi quá trình tự thực thi. Tại thời điểm này, nếu toàn bộ quá trình hoặc bộ nhớ bị dừng lại và bị hủy, ít nhất sẽ có một cái gì đó để phân tích.

#### *1.4.21. Parsing Memory Dump Contents*

Các công cụ được mô tả trong DFRWS 2005 Memory Challenge đã sử dụng một phương pháp để phân tích nội dung bộ nhớ của việc định vị và liệt kê danh sách quy trình hoạt động, sử dụng các giá trị / giá trị cụ thể (xuất phát từ các tệp hệ thống) để xác định phần đầu của danh sách và sau đó đi qua liên kết đôi liệt kê cho đến khi tất cả các quy trình hoạt động đã được xác định. Vị trí của phần bù cho phần đầu của danh sách quy trình hoạt động được lấy từ một trong các tệp hệ thống quan trọng, ntoskrnl.exe.

Andreas Schuster đã thực hiện một cách tiếp cận khác trong kịch bản Perl của mình, được gọi là ptfinder.pl. Ý tưởng của anh là đưa ra một cách tiếp cận mạnh mẽ cho vấn đề mà xác định các đặc điểm cụ thể của các quá trình trong bộ nhớ và sau đó liệt kê các khối EProcess cũng như các thông tin khác về các quy

trình dựa trên các đặc điểm đó. Andreas bắt đầu cách tiếp cận của mình bằng cách liệt kê cấu trúc của DISPATCHER\_HEADER, được đặt ở vị trí bù 0 cho mỗi khối EProcess (thực ra, nó nằm trong cấu trúc được gọi là khối KProcess). Sử dụng LiveKD, chúng tôi thấy rằng cấu trúc được liệt kê từ hệ thống Windows 2000 có các yếu tố sau:

```
+0x000 Header : struct _DISPATCHER_HEADER, 6 elements, 0x10 bytes
+0x000 Type      : UChar
+0x001 Absolute  : UChar
+0x002 Size      : UChar
+0x003 Inserted  : UChar
+0x004 SignalState : Int4B
```

Tóm lại, Andreas phát hiện ra rằng một số yếu tố cho DISPATCHER\_HEADER là nhất quán trong tất cả các quy trình trên hệ thống. Ông đã kiểm tra các phần tử DISPATCHER\_HEADER cho các quy trình (và luồng) trên các hệ thống từ Windows 2000 cho đến các bản beta đầu tiên của Vista và thấy rằng giá trị Loại vẫn nhất quán trên từng phiên bản của hệ điều hành. Ông cũng nhận thấy rằng giá trị Kích thước vẫn nhất quán trong các phiên bản khác nhau của hệ điều hành (ví dụ: tất cả các quy trình trên Windows 2000 hoặc XP có cùng giá trị Kích thước) nhưng đã thay đổi giữa các phiên bản đó (ví dụ: đối với Windows 2000, giá trị Kích thước là 0x1b, nhưng đối với các phiên bản đầu của Vista, nó là 0x20).

Sử dụng thông tin này cũng như tổng kích thước của cấu trúc và cách cấu trúc có thể bị phá vỡ, Andreas đã viết tập lệnh ptfinder.pl Perl của mình, để liệt kê các quy trình và luồng nằm trong kết xuất bộ nhớ. Tại hội nghị DFRWS 2006, ông cũng đã trình bày một bài báo "Searching for processes and threads in Microsoft Windows memory dumps" ([www.dfrws.org/2006/proceedings/2-Schuster.pdf](http://www.dfrws.org/2006/proceedings/2-Schuster.pdf)), không chỉ đề cập đến các cấu trúc dữ liệu tạo ra lên các quy trình và luồng nhưng cũng có nhiều quy tắc khác nhau để xác định xem những gì được tìm thấy là một cấu trúc hợp pháp hay chỉ là một bó byte trong một tệp.

### **Chú ý:**

Vào mùa thu năm 2006, Richard McQuown đã kết hợp một giao diện GUI cho các công cụ Andreas Schuster tựa PTFinder. Các công cụ PTFinder là các tập lệnh Perl và yêu cầu trình thông dịch Perl được cài đặt trên một hệ thống để chạy chúng. (Perl được cài đặt theo mặc định trên nhiều bản phân phối Linux và có sẵn miễn phí cho các nền tảng Windows từ ActiveState.com.) Công cụ của Richard không chỉ có thể phát hiện hệ điều hành của kết xuất RAM (thay vì người dùng nhập thủ công) bằng mã I ' sẽ thảo luận sau trong chương này, nhưng nó cũng có thể cung cấp một biểu diễn đồ họa của đầu ra. PTFinderFE có một số ứng dụng thú vị, đặc biệt liên quan đến trực quan hóa.

Vào mùa xuân năm 2006, tôi đã viết một số công cụ của riêng mình để hỗ trợ phân tích cú pháp thông qua các tệp kết xuất RAM của Windows. Bởi vì các ví dụ hiện có tại thời điểm đó là các kết xuất cho các hệ thống Windows 2000 có sẵn từ DFRWS 2005 Memory challenge, tôi đã tập trung nỗ lực ban đầu của mình vào việc sản xuất mã hoạt động cho nền tảng đó. Điều này cho phép tôi giải quyết các vấn đề khác nhau trong phát triển mã mà không bị cuốn vào vô số sự khác biệt giữa các phiên bản khác nhau của hệ điều hành Windows. Kết quả là bốn tập lệnh Perl riêng biệt, mỗi tập lệnh chạy từ dòng lệnh. Tất cả các tập lệnh này được cung cấp trên DVD đi kèm và chúng tôi sẽ thảo luận về chúng ở đây.

### **Chú ý:**

Các công cụ sau (lsproc.pl, lspd.pl, lspj.pl và lspm.pl) được thiết kế để chỉ được sử dụng với các tệp bộ nhớ Windows 2000. Như chúng tôi đã thảo luận cho

đến nay, có những thay đổi đáng kể trong định dạng cấu trúc EProcess giữa các phiên bản khác nhau của Windows (2000, XP, 2003, Vista, v.v.). Như vậy, công việc quan trọng cần phải được thực hiện để tạo ra một ứng dụng duy nhất cho phép bạn phân tích cú pháp bộ nhớ từ tất cả các phiên bản.

#### 1.4.22. *Lsproc.pl*

LSproc, viết tắt của các quy trình danh sách, tương tự như Andreasfinder ptfinder.pl; tuy nhiên, lsproc.pl định vị quy trình nhưng không phải chủ đề. Lsproc.pl nhận một đối số duy nhất, đường dẫn và tên cho tệp kết xuất RAM:

***c:\perl\memory>lsproc.pl d:\dumps\drfws1-mem.dmp***

Đầu ra của lsproc.pl xuất hiện tại bàn điều khiển (nghĩa là STDOUT) trong sáu cột: từ Proc (Tôi đã dự đoán thêm các chủ đề vào một ngày sau đó), định danh quy trình cha mẹ (PPID), định danh quy trình (PID), Tên của quá trình, phần bù của cấu trúc quy trình trong tệp kết xuất và thời gian tạo của quy trình. Đây là một đoạn trích của lsproc. đầu ra pl:

Proc	820	324	helix.exe	0x00306020	Sun	Jun 5	14:09:27	2005
Proc	0	0	Idle	0x0046d160				
Proc	600	668	UMGR32.EXE	0x0095f020	Sun	Jun 5	00:55:08	2005
Proc	324	1112	cmd2k.exe	0x00dcc020	Sun	Jun 5	14:14:25	2005
Proc	668	784	dfrws2005.exe (x)	0x00e1fb60	Sun	Jun 5	01:00:53	2005
Proc	156	176	winlogon.exe	0x01045d60	Sun	Jun 5	00:32:44	2005
Proc	156	176	winlogon.exe	0x01048140	Sat	Jun 4	23:36:31	2005

Proc	144	164	winlogon.exe	0x0104ca00	Fri	Jun 3	01:25:54	2005
Proc	156	180	csrss.exe	0x01286480	Sun	Jun 5	00:32:43	2005
Proc	144	168	csrss.exe	0x01297b40	Fri	Jun 3	01:25:53	2005
Proc	8	156	smss.exe	0x012b62c0	Sun	Jun 5	00:32:40	2005
Proc	0	8	System	0x0141dc60				
Proc	668	784	dfrws2005.exe (x)	0x016a9b60	Sun	Jun 5	01:00:53	2005
Proc	1112	1152	dd.exe (x)	0x019d1980	Sun	Jun 5	14:14:38	2005
Proc	228	592	dfrws2005.exe	0x02138640	Sun	Jun 5	01:00:53	2005
Proc	820	1076	cmd.exe	0x02138c40	Sun	Jun 5	00:35:18	2005
Proc	240	788	metasploit.exe (x)	0x02686cc0	Sun	Jun 5	00:38:37	2005
Proc	820	964	Apoint.exe	0x02b84400	Sun	Jun 5	00:33:57	2005
Proc	820	972	HKserv.exe	0x02bf86e0	Sun	Jun 5	00:33:57	2005
Proc	820	988	DragDrop.exe	0x02c46020	Sun	Jun 5	00:33:57	2005
Proc	820	1008	alogserv.exe	0x02e7ea20	Sun	Jun 5	00:33:57	2005
Proc	820	972	HKserv.exe	0x02f806e0	Sun	Jun 5	00:33:57	2005
Proc	820	1012	tgcmd.exe	0x030826a0	Sun	Jun 5	00:33:58	2005
Proc	176	800	userinit.exe (x)	0x03e35020	Sun	Jun 5	00:33:52	2005
Proc	800	820	Explorer.Exe	0x03e35ae0	Sun	Jun 5	00:33:53	2005
Proc	820	1048	PcfMgr.exe	0x040b4660	Sun	Jun 5	00:34:01	2005

Quá trình đầu tiên được liệt kê trong đầu ra *lsproc.pl* là *helix.exe*. Theo thông tin được cung cấp tại trang web DFRWS 2005 Memory Challenge, các tiện ích trên đĩa CD Helix Live đã được sử dụng để thu được kết xuất bộ nhớ.

Danh sách trước chỉ hiển thị một đoạn trích của đầu ra *lsproc.pl*. Tổng cộng 45 các quy trình được đặt trong tệp kết xuất bộ nhớ. Bạn sẽ nhận thấy trong đầu ra rằng một số quy trình có (x) sau tên quy trình. Điều này chỉ ra rằng các quy trình đã thoát.

### Ghi chú:

Nhìn kỹ, bạn sẽ nhận thấy một số điều thú vị về đầu ra *lsproc.pl*. Một là quá trình *csrss.exe* (PID = 168) có ngày tạo dường như sớm hơn một hoặc hai ngày so với các quy trình được liệt kê khác. Nhìn kỹ hơn nữa, bạn sẽ thấy một cái gì đó tương tự cho hai quá trình *winlogon.exe* (PID = 164 và 176). Andreas Schuster cũng nhận thấy những điều này và theo một mục về tính bền vững dữ

liệu trong blog của anh ấy ([http://computer.forensikblog.de/en/2006/04/continance\\_ENC\\_the\\_boot\\_process.html](http://computer.forensikblog.de/en/2006/04/continance_ENC_the_boot_process.html)), thời gian khởi động hệ thống cho tệp kết xuất đã được xác định đến Chủ nhật, ngày 5 tháng 1 năm 2005, vào khoảng 00:32:27. Vậy, những quá trình này đến từ đâu?

Như Andreas chỉ ra trong blog của mình, không có thông tin chính xác hơn về trạng thái của hệ thống kiểm tra trước khi thu thập dữ liệu cho Thử thách bộ nhớ, rất khó để phát triển sự hiểu biết đầy đủ về vấn đề này. Tuy nhiên, các thông số kỹ thuật của hệ thống kiểm tra đã được biết và ghi lại, và lưu ý rằng hệ thống đã gặp sự cố trong quá trình thu thập dữ liệu. Hoàn toàn có thể là dữ liệu sống sót sau khi khởi động lại. Dường như không có bất kỳ thông số kỹ thuật nào yêu cầu rằng khi một hệ thống Windows tắt hoặc gặp sự cố, các nội dung của bộ nhớ vật lý sẽ bị xóa hoặc xóa theo một cách nào đó. Sau đó, có thể nội dung của bộ nhớ vật lý vẫn ở trạng thái trước đó và nếu chúng không bị ghi đè khi hệ thống được khởi động lại, dữ liệu vẫn có sẵn để phân tích. Nhiều BIOS các phiên bản có tính năng ghi đè bộ nhớ trong khi khởi động như một phần của kiểm tra RAM, nhưng tính năng này thường bị vô hiệu hóa để tăng tốc quá trình khởi động. Đây chắc chắn là một lĩnh vực đòi hỏi phải nghiên cứu thêm. Như Andreas tuyên bố ([http://computer.forensikblog.de/en/2006/04/data\\_lifetime.html](http://computer.forensikblog.de/en/2006/04/data_lifetime.html)), lĩnh vực nghiên cứu này có một tương lai tươi sáng.

#### 1.4.23. *Lspd.pl*

Lspd.pl là một tập lệnh Perl sẽ cho phép bạn liệt kê các chi tiết của quy trình. Giống như các công cụ khác mà chúng ta sẽ thảo luận, lspd.pl là một tập lệnh Perl dòng lệnh dựa trên đầu ra của lsproc.pl để có được thông tin của nó. Cụ thể, lspd.pl có hai đối số: đường dẫn đầy đủ của tệp kết xuất và phần bù vật lý của quá trình mà bạn có thể quan tâm (vật lý bù của quá trình trong bộ nhớ được lấy từ đầu ra

lsproc.pl). Mặc dù lsproc.pl mất một chút thời gian để phân tích nội dung của tệp kết xuất, lspd.pl nhanh hơn nhiều, vì bạn có thể nói chính xác nơi cần đi trong tệp để liệt kê thông tin của nó.

Hãy cùng xem một quy trình cụ thể. Trong trường hợp này, chúng ta sẽ xem xét dd.exe, quy trình với PID 284. Dòng lệnh sử dụng lspd.pl để nhận thông tin chi tiết về quy trình này là:

***c:\perl\memory>lspd.pl d:\dumps\dfrrws1-mem.dmp 0x0414dd60***

**Ghi chú:**

Đầu ra lsproc.pl vừa hiển thị là một đoạn trích của toàn bộ đầu ra; Tôi đã liệt kê toàn bộ đầu ra đơn giản vì đoạn trích minh họa đủ thông tin để tôi đưa ra quan điểm của mình. Tuy nhiên, quá trình được tham chiếu trong lspd.pl dòng lệnh (tức là, tại offset 0x0414dd60) không được liệt kê trong đoạn trích đó, mặc dù nó được hiển thị trong toàn bộ đầu ra của lsproc.pl.

Process Name	: dd.exe
PID	: 284
Parent PID	: 1112
TFLINK	: 0xff2401c4
TBLINK	: 0xff2401c4
FLINK	: 0x8046b980
BLINK	: 0xff1190c0
SubSystem	: 4.0
Exit Status	: 259
Create Time	: Sun Jun 5 14:53:42 2005
Exit Called	: 0
DTB	: 0x01d9e000



ObjTable	:	0xff158708 (0x00eb6708)
PEB	:	0x7ffdf000 (0x02c2d000)
InheritedAddressSpace	:	0
ReadImageFileExecutionOptions	:	0
BeingDebugged	:	0
CSDVersion	:	Service Pack 1
Mutant	=	0xffffffff
Img Base Addr	=	0x00400000 (0x00fee000)
PEB_LDR_DATA	=	0x00131e90 (0x03a1ee90)
Params	=	0x00020000 (0x03a11000)

### Ghi chú:

Trước đó trong chương này, tôi đã đề cập rằng danh sách các quy trình hoạt động trên một hệ thống trực tiếp được duy trì trong một danh sách liên kết đôi. Các giá trị nhấp nháy và nhấp nháy nhìn thấy trong đầu ra lspd.pl trước đó lần lượt là các giá trị trỏ đến các quá trình tiếp theo và trước đó. Như được hiển thị trong đầu ra của lspd.pl, các con trỏ này là tới các địa chỉ trong bộ nhớ, không phải là địa chỉ vật lý hoặc offset trong tệp kết xuất.

Lspd.pl cũng theo dõi các con trỏ được cung cấp bởi cấu trúc EProcess để thu thập dữ liệu khác. Ví dụ: chúng ta cũng có thể thấy đường dẫn đến hình ảnh thực thi và dòng lệnh được sử dụng để khởi chạy quy trình (được thêm đậm để nhấn mạnh):

```

Current Directory Path = E:\Shells\
DllPath                = E:\Acquisition\FAU\.;C:\WINNT\System32;C:\WINNT\system;
                        C:\WINNT;E:\Acquisition\FAU\;E:\Acquisition\GNU\;
                        E:\Acquisition\CYGIN\;E:\IR\bin\;E:\IR\WFT;E:\IR\
                        windbg\;E:\IR\Foundstone\;E:\IR\Cygwin;E:\IR\
                        somarsoft\;E:\IR\sysinternals\;E:\IR\ntsecurity\;
                        E:\IR\perl\;E:\Static-Binaries\gnu_utils_win32\;C:\WINNT\
                        system32;C:\WINNT;C:\WINNT\System32\Wbem

ImagePathName          = E:\Acquisition\FAU\dd.exe
Command Line           = ..\Acquisition\FAU\dd.exe if=\\.\PhysicalMemory of=F:\
                        intrusion2005\physicalmemory.dd conv=noerror --md5sum
                        --verifymd5 --md5out=F:\intrusion2005\physicalmemory.dd.
                        md5 --log=F:\intrusion2005\audit.log

Environment Offset     = 0x00000000 (0x00000000)
Window Title           = ..\Acquisition\FAU\dd.exe if=\\.\PhysicalMemory of=F:\
                        intrusion2005\physicalmemory.dd conv=noerror --md5sum
                        --verifymd5 --md5out=F:\intrusion2005\physicalmemory.dd.
                        md5 --log=F:\intrusion2005\audit.log

Desktop Name           = WinSta0\Default

```

Lspd.pl cũng lấy ra một danh sách tên của các mô-đun (DLL) khác nhau được sử dụng bởi quy trình và bất kỳ xử lý có sẵn nào (xử lý tệp, v.v.) mà nó có thể tìm thấy trong bộ nhớ. Ví dụ: lspd.pl thấy rằng dd.exe có tệp sau xử lý mở:

### ***Type : File***

***Name = \intrusion2005\audit.log***

Như bạn có thể thấy từ dòng lệnh trước, tệp \intrusion\audit.log nằm trên ổ F: \ và là tệp đầu ra cho nhật ký hoạt động được tạo bởi dd.exe, giải thích tại sao nó sẽ được liệt kê dưới dạng một xử lý tệp tin mở được sử dụng bởi quá trình. Sử dụng thông tin này có nguồn gốc từ các quy trình khác, bạn có thể hiểu được các tệp bạn nên quan tâm trong quá trình phân tích. Trong trường hợp cụ thể này, bạn có thể giả sử rằng ổ E: \ được liệt kê trong ImagePathName là ổ đĩa CD-ROM, vì Helix có thể được chạy từ CD. Bạn có thể xác nhận điều này bằng cách kiểm tra các giá trị Registry trong một hình ảnh của hệ thống đang được đề cập (tuy nhiên hình ảnh hệ thống không được cung cấp như một phần của thách thức bộ nhớ). Bạn cũng có thể sử dụng thông tin tương tự để tìm hiểu thêm một chút về ổ F: \. Tôi sẽ bao gồm điều này thông tin trong chương 4.

Cuối cùng, một điều khác mà `lspd.pl` sẽ làm là đi đến vị trí được chỉ ra bởi giá trị `Addr` của `Image Base` (một khi nó đã được dịch từ một địa chỉ ảo sang phần bù vật lý trong tệp kết xuất bộ nhớ) và kiểm tra xem hình ảnh thực thi hợp lệ được đặt tại địa chỉ đó. Kiểm tra này rất đơn giản; tất cả những gì nó làm là đọc hai byte đầu tiên bắt đầu tại địa chỉ được dịch để xem liệu chúng có phải là `MZ` hay không. Hai byte này không phải là một kiểm tra chính xác, nhưng các tệp PE (các tệp có `.exe`, `.Ocs`, `.sys` và các phần mở rộng tương tự) bắt đầu bằng tên viết tắt của Mark Zbikowski, một trong những kiến trúc sư đầu tiên của MS-DOS và Windows NT. Định dạng của tệp PE và tiêu đề của nó được đề cập chi tiết hơn trong Chương 6.

### **Chú ý:**

Nếu bạn kết xuất nội dung của bộ nhớ vật lý từ hệ thống Windows 2000 hoặc XP bằng `winen.exe` và bạn có khóa EnCase được cấp phép, bạn có thể phân tích thông tin xử lý từ kết xuất bộ nhớ bằng cách sử dụng `EnScripts` do TK\_Lane viết và có sẵn thông qua "EDD and Forensics".

#### *1.4.24. Volatility Framework*

Aaron Walters cung cấp một số thông tin có giá trị về Volatility Framework trong bài thuyết trình OMFW của mình, có sẵn từ [https://www.volatilesystems.com/volatility/omfw/Walters\\_OMFW\\_2008.pdf](https://www.volatilesystems.com/volatility/omfw/Walters_OMFW_2008.pdf).

Tệp `readme.txt` là một phần của Volatility Framework (Phiên bản 1.3 beta tại thời điểm viết bài này) cung cấp rất nhiều thông tin về cách sử dụng Volatility, loại lệnh và khả năng nào có sẵn, cũng như các ví dụ về cách thức để khởi chạy các lệnh khác nhau. Aaron đã thiết kế Biến động để sử dụng một số lệnh thường được sử dụng trong các hoạt động ứng phó sự cố; ví dụ, để có được danh sách các tiến trình đang chạy từ kết xuất bộ nhớ, Biến động sử dụng `pslist`. Trước khi sử dụng

Biến động, đảm bảo đọc qua tệp readme.txt để xem loại thông tin nào có thể được truy xuất từ kết xuất bộ nhớ Windows XP SP2 hoặc SP3.

Để minh họa loại thông tin nào có sẵn từ kết xuất bộ nhớ thô, kiểu dd, hãy để xem một ví dụ; trong trường hợp này, kết xuất bộ nhớ 512 MB từ máy tính xách tay Windows XP SP2. Chúng ta có thể bắt đầu bằng cách lấy một số thông tin cơ bản về kết xuất bộ nhớ bằng lệnh nhận dạng:

```
D:\Volatility>python volatility ident -f d:\hacking\xp-laptop1.img
Image Name      : d:\hacking\xp-laptop1.img
Image Type      : Service Pack 2
VM Type         : nopae
DTB             : 0x39000
Datetime        : Sat Jun 25 12:58:47 2005
```

Đây có thể là thông tin rất hữu ích trong việc ghi lại phân tích của chúng tôi về kết xuất bộ nhớ, vì trong một số trường hợp, chúng tôi có thể không có quyền truy cập vào thông tin nhận dạng như một phần của tài liệu của chúng tôi. Sử dụng lệnh pslist, chúng tôi có thể truy xuất danh sách quy trình hoạt động từ kết xuất bộ nhớ theo định dạng tương tự như những gì chúng tôi đã từng thấy khi chạy pslist.exe trên hệ thống trực tiếp:

```
D:\Volatility>python volatility pslist -f d:\hacking\xp-laptop1.img
```

Name	Pid	PPid	Thds	Hnds	Time			
System	4	0	61	1140	Thu	Jan	01	00:00:00 1970
smss.exe	448	4	3	21	Sat	Jun	25	16:47:28 2005
csrss.exe	504	448	12	596	Sat	Jun	25	16:47:30 2005
winlogon.exe	528	448	21	508	Sat	Jun	25	16:47:31 2005
services.exe	580	528	18	401	Sat	Jun	25	16:47:31 2005
lsass.exe	592	528	21	374	Sat	Jun	25	16:47:31 2005
svchost.exe	740	580	17	198	Sat	Jun	25	16:47:32 2005
svchost.exe	800	580	10	302	Sat	Jun	25	16:47:33 2005
svchost.exe	840	580	83	1589	Sat	Jun	25	16:47:33 2005
Smc.exe	876	580	22	423	Sat	Jun	25	16:47:33 2005
svchost.exe	984	580	6	90	Sat	Jun	25	16:47:35 2005
svchost.exe	1024	580	15	207	Sat	Jun	25	16:47:35 2005
spoolsv.exe	1224	580	12	136	Sat	Jun	25	16:47:39 2005
ssonsvr.exe	1632	1580	1	24	Sat	Jun	25	16:47:46 2005
explorer.exe	1812	1764	22	553	Sat	Jun	25	16:47:47 2005
Directcd.exe	1936	1812	4	40	Sat	Jun	25	16:47:48 2005
TaskSwitch.exe	1952	1812	1	21	Sat	Jun	25	16:47:48 2005
Fast.exe	1960	1812	1	22	Sat	Jun	25	16:47:48 2005
VPTray.exe	1980	1812	2	89	Sat	Jun	25	16:47:49 2005
atiptaxx.exe	2040	1812	1	51	Sat	Jun	25	16:47:49 2005
jusched.exe	188	1812	1	22	Sat	Jun	25	16:47:49 2005
EM_EXEC.exe	224	112	2	74	Sat	Jun	25	16:47:50 2005
ati2evxx.exe	432	580	4	38	Sat	Jun	25	16:47:55 2005
Crypserve.exe	688	580	3	34	Sat	Jun	25	16:47:55 2005
DefWatch.exe	864	580	3	27	Sat	Jun	25	16:47:55 2005
msdtc.exe	1076	580	14	166	Sat	Jun	25	16:47:55 2005
Rtvscan.exe	1304	580	37	300	Sat	Jun	25	16:47:58 2005
tcpsvcs.exe	1400	580	2	94	Sat	Jun	25	16:47:58 2005
snmp.exe	1424	580	5	192	Sat	Jun	25	16:47:58 2005
svchost.exe	1484	580	6	119	Sat	Jun	25	16:47:59 2005
wdfmgr.exe	1548	580	4	65	Sat	Jun	25	16:47:59 2005
Fast.exe	1700	580	2	32	Sat	Jun	25	16:48:01 2005
mqsvc.exe	1948	580	23	205	Sat	Jun	25	16:48:02 2005
mqtgsvc.exe	2536	580	9	119	Sat	Jun	25	16:48:05 2005
alg.exe	2868	580	6	108	Sat	Jun	25	16:48:11 2005

wuauclt.exe	2424	840	4	160	Sat	Jun	25	16:49:21	2005
firefox.exe	2160	1812	6	182	Sat	Jun	25	16:49:22	2005
PluckSvr.exe	944	740	9	227	Sat	Jun	25	16:51:00	2005
iexplore.exe	2392	1812	9	365	Sat	Jun	25	16:51:02	2005
PluckTray.exe	2740	944	3	105	Sat	Jun	25	16:51:10	2005
PluckUpdater.ex	3076	1812	0	-1	Sat	Jun	25	16:51:15	2005
PluckUpdater.ex	1916	944	0	-1	Sat	Jun	25	16:51:40	2005
PluckTray.exe	3256	1812	0	-1	Sat	Jun	25	16:54:28	2005
cmd.exe	2624	1812	1	29	Sat	Jun	25	16:57:36	2005
wmiprvse.exe	4080	740	7	0	Sat	Jun	25	16:57:53	2005
PluckTray.exe	3100	1812	0	-1	Sat	Jun	25	16:57:59	2005
dd.exe	4012	2624	1	22	Sat	Jun	25	16:58:46	2005

Chúng ta có thể chạy các lệnh tương tự để lấy thông tin về tất cả các đối tượng trong kết xuất bộ nhớ, bao gồm các tiến trình đã thoát, sử dụng lệnh *psscan* hoặc *psscan2*. Các lệnh để lấy thông tin về tất cả các đối tượng (kết nối mạng, quy trình, v.v.) chậm hơn, vì họ sử dụng phương pháp quét tuyến tính để chạy hoàn toàn thông qua tệp kết xuất bộ nhớ, kiểm tra tất cả các đối tượng có thể, thay vì sử dụng các thông số cụ thể do hệ điều hành cung cấp (xem phần thảo luận về LiveKD trước đó trong chương này). Một trong những điều hữu ích hơn mà hầu hết các nhà phân tích tìm đến khi gặp một vụ xâm nhập hoặc thỏa hiệp là kết nối mạng. Bạn có thể lấy danh sách các kết nối mạng đang hoạt động (tương tự như sử dụng lệnh *netstat -ano*) từ kết xuất bộ nhớ bằng các kết nối lệnh, như sau:

```
D:\Volatility>python volatility connections -f d:\hacking\xp-laptop1.img
```

Local Address	Remote Address	Pid
127.0.0.1:1056	127.0.0.1:1055	2160
127.0.0.1:1055	127.0.0.1:1056	2160
192.168.2.7:1077	64.62.243.144:80	2392
192.168.2.7:1082	205.161.7.134:80	2392
192.168.2.7:1066	199.239.137.200:80	2392

Bước tiếp theo, bạn có thể quét toàn bộ tệp kết xuất bộ nhớ để tìm chỉ dẫn về các đối tượng kết nối mạng, đặc biệt tìm kiếm các kết nối mạng có thể có đã bị đóng tại thời điểm kết xuất bộ nhớ:

```
D:\Volatility>python volatility connscan2 -f d:\hacking\xp-laptop1.img
```

Local Address	Remote Address	Pid
-----	-----	-----
192.168.2.7:1115	207.126.123.29:80	1916
3.0.48.2:17985	66.179.81.245:20084	4287933200
192.168.2.7:1164	66.179.81.247:80	944
192.168.2.7:1082	205.161.7.134:80	2392
192.168.2.7:1086	199.239.137.200:80	1916
192.168.2.7:1162	170.224.8.51:80	1916
127.0.0.1:1055	127.0.0.1:1056	2160
192.168.2.7:1116	66.161.12.81:80	1916
192.168.2.7:1161	66.135.211.87:443	1916
192.168.2.7:1091	209.73.26.183:80	1916
192.168.2.7:1151	66.150.96.111:80	1916
192.168.2.7:1077	64.62.243.144:80	2392
192.168.2.7:1066	199.239.137.200:80	2392
192.168.2.7:1157	66.151.149.10:80	1916
192.168.2.7:1091	209.73.26.183:80	1916
192.168.2.7:1115	207.126.123.29:80	1916
192.168.2.7:1155	66.35.250.150:80	1916
127.0.0.1:1056	127.0.0.1:1055	2160
192.168.2.7:1115	207.126.123.29:80	1916
192.168.2.7:1155	66.35.250.150:80	1916

Volatility là một framework mã nguồn mở mạnh mẽ, cho phép người khác mở rộng khả năng của nó bằng cách phát triển các mô-đun bổ sung (kiến thức về lập trình Python là một yêu cầu quan trọng). Brendan Dolan-Gavitt (a.k.a. Moyix) đã tạo ra một mô-đun tìm kiếm động cho tin nhắn Window (<http://moyix.blogspot.com/2008/09/window-messages-as-forensic-resource.html>), đó là những sự kiện khác nhau được tạo bởi các ứng dụng GUI của

Windows và được xử lý bởi thông báo hàng đợi. Như Brendan chỉ ra, một ứng tòi có thể không xử lý được tin nhắn riêng rất ; trong trường hợp này, bạn có thể tìm thấy dấu vết của những tin nhắn đó vẫn hiển thị trong bãi chứa bộ nhớ. Thông tin này có thể hữu ích trong quá trình phân tích.

### **Chú ý:**

Brendan cũng đã tạo ra các Volatility plug-ins để truy cập dữ liệu tìm thấy được trong các bãi chứa bộ nhớ Windows (tại <http://moyix.blogspot.com/2009/01/memory-registry-tools.html> và các bản cập nhật cho mã ở <http://moyix.blogspot.com/2009/01/regology-code-updates.html>). Toàn bộ blog của Brendan (<http://forensiczone.blogspot.com/2009/01/USE-volatility-1.html>),

Richard McQuown đã trình diễn bằng cách sử dụng các mô-đun này để trích xuất mật khẩu từ tệp hive trong Trình quản lý tài khoản bảo mật (SAM)(nằm trong bộ nhớ) để có thể bẻ khóa những mật khẩu đó bằng công cụ của mình. Để sử dụng các mô-đun Volatility của Brendan và trích xuất mật khẩu từ tổ hợp các tệp nằm trong bộ nhớ, bạn phải cài đặt các mô-đun PyCrypto (có sẵn như các tệp nhị phân Windows dựng sẵn từ [www.voidspace.org.uk/python/modules.shtml#pycrypto](http://www.voidspace.org.uk/python/modules.shtml#pycrypto)).

Ngoài ra, Jesse Kornblum đã sản xuất hai mô-đun: đáng nghi ngờ, có vẻ đáng nghi ngờ các mục trong dòng lệnh xử lý và cryptoscan, tìm cụm mật khẩu TrueCrypt. Mô-đun cuối cùng này cực kỳ có ích cho nhà phân tích, như TrueCrypt ([www.truecrypt.org/](http://www.truecrypt.org/)) là một ứng dụng mạnh mẽ, mặc dù miễn phí, có thể được sử dụng để đĩa cứng và đĩa. Tính Volatility hoạt động hiệu quả hơn là bộ nhớ thô. Nhờ sự nỗ lực của Matthieu Suiche ([www.msuiche.net/](http://www.msuiche.net/)), Volatility bao gồm khả năng phân tích cú pháp tập tin đóng băng, là rất tốt. Điều này bắt đầu giống như Dự án Sandman, và sau đó trở thành một phần không



thể thiếu của Framework Volatility. Vào tháng 12 năm 2008, Matthieu đã phát hành độc lập, phiên bản nguồn đóng (alpha) của Framework Hibernation shell, được gọi là hibrshell ([www.msuiche.net / hibrshell /](http://www.msuiche.net/hibrshell/)). Phiên bản hibrshell này hoạt động với các tệp đóng băng từ các hệ thống Windows XP, 2003, Vista và 2008.

Bất kể framework nào được sử dụng để phân tích, tệp đóng băng cung cấp một bên phản hồi hoặc bên phân tích với một số tùy chọn mà trước đây không có. Đầu tiên, tệp tin đóng băng có thể được sử dụng làm dữ liệu lịch sử, cung cấp thông tin về hệ thống trực tiếp, trạng thái hoạt động tại một thời điểm trước đó. Cái này có thể cực kỳ có giá trị trong phân tích phần mềm độc hại, cũng như để giúp xác định một mốc thời gian xâm nhập, đặc biệt nếu nhà phân tích cũng có kết xuất bộ nhớ hiện tại để phân tích. Trong trường hợp các công cụ được đề cập trước đó (ví dụ: mdd.exe, v.v.) không thể sử dụng để kết xuất nội dung của bộ nhớ vật lý từ một hệ thống, bộ phản hồi có thể buộc hệ thống đóng băng để tạo ra một bãi chứa bộ nhớ mà sau đó có thể được phân tích.

Tính không ổn định cũng có thể phân tích cú pháp các tệp kết xuất sự cố, cũng như chuyển đổi bộ nhớ thô, kiểu dump để sụp đổ định dạng dump nhằm giúp nhà phân tích có thể sử dụng các Debugging Tools của Microsoft. Bây giờ, rõ ràng là Framework Volatility cung cấp khả năng mạnh mẽ và số lượng thông tin mà nhà phân tích có thể lấy từ bộ nhớ xuống. Để giúp tương quan một số dữ liệu có thể được truy xuất bằng Volatility, Jamie Levy đã viết một tập lệnh Perl có tên là *vol2html.pl* (<http://gleeda.blogspot.com/2008/11/vol2html-perlscript.html>). Kịch bản nhận đầu ra của các lệnh *pslist*, tệp *,dlllist* Volatility và tương quan giữa chúng với một báo cáo HTML. Tương tự như *listdlls.exe* quen thuộc có sẵn từ Microsoft (Sysinternals), lệnh *dlllist* Volatility bao gồm dòng lệnh quy trình như một phần của đầu ra; dòng lệnh này cũng xuất hiện trong đầu ra

HTML của *vol2html.pl*.

Ví dụ về các tệp kết xuất bộ nhớ Windows XP có sẵn như là một phần của DFRWS 2008 Forensic Rodeo ([www.dfrws.org/2008/rodeo.shtml](http://www.dfrws.org/2008/rodeo.shtml)), cũng như từ Viện quốc gia về Tiêu chuẩn và Công nghệ.

Michael Hale Ligh cung cấp hai bài đăng trên blog mô tả cách sử dụng Framework Volatility có hiệu quả lớn, đặc biệt liên quan đến phân tích phần mềm độc hại; phục hồi các mã nhị phân CoreFlood với tính dễ bay hơi (<http://mnin.blogspot.com/2008/11/recoveringcoreflood-binaries-with.html>) và xác định vị trí các tệp DLL kẹp ẩn (kiểu VAD) (<http://mnin.blogspot.com/2008/11/locating-hidden-clampi-dlls-vad-style.html>). Cả hai bài viết trên blog đều cung cấp những ví dụ tuyệt vời về cách Framework Volatility có thể tối đa hóa khả năng của nhà phân tích.

#### 1.4.25. Memoryze

Công cụ bộ nhớ Mandiant cung cấp cho nhà phân tích khả năng phân tích và phân tích bộ nhớ chổi bỏ từ một số phiên bản của Windows. Để cài đặt Memoryze, tải xuống tệp MSI từ trang web Mandiant (đã đề cập trước đó trong chương này) và cài đặt nó. Chọn cài đặt nó trong *D: \ Mandiant Folder*. Sau đó, để cài đặt trình xem kiểm toán, hãy tải xuống kho lưu trữ đã nén và chắc chắn mà bạn đã tải xuống các phần phụ thuộc (ví dụ: phần mở rộng GUI Python 2.5 hoặc 2.6, wxPython) dưới dạng được mô tả tại trang web Mandiant (nếu bạn đã cài đặt và dùng thử Volatility, bạn đã cài đặt Python). Giải nén các tệp Audit Viewer vào thư mục *D: \ Mandiant \ AV*. Để bắt đầu việc sử dụng Memoryze và Audit Viewer, bắt đầu bằng cách chọn bộ nhớ kết xuất từ hệ thống Windows 2003: boomer-

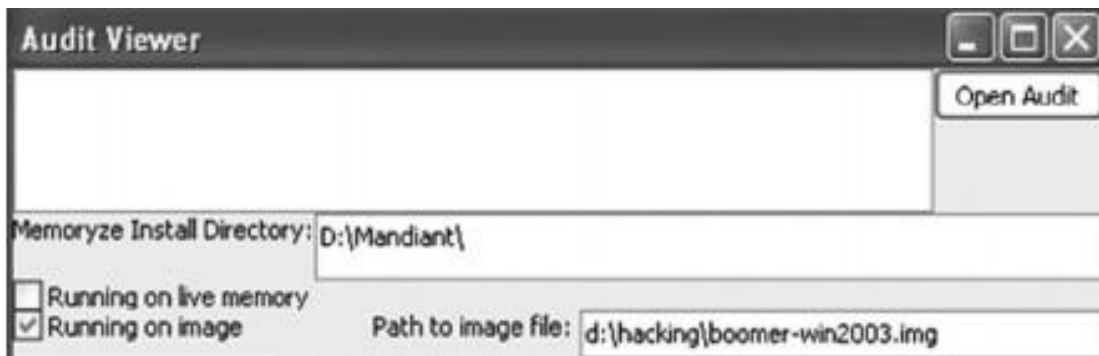
win2003.img. Điều đầu tiên cần phải làm để phân tích kết xuất bộ nhớ này là chạy Memoryze với nó để trích xuất nhiều dữ liệu khác nhau:

***D:\mandiant>process.bat -input d:\hacking\boomer-win2003.img -ports true -handles true -sections true***

Lệnh trên yêu cầu Memoryze phân tích thông tin quy trình từ kết xuất bộ nhớ và nhận các cổng, tay cầm và các phần bộ nhớ (chọn không nhận các chuỗi từ mỗi quy trình) cho các quy trình trong danh sách quy trình hoạt động. Phạm vi đầy đủ của tùy chọn sử dụng cho *process.bat* bao gồm:

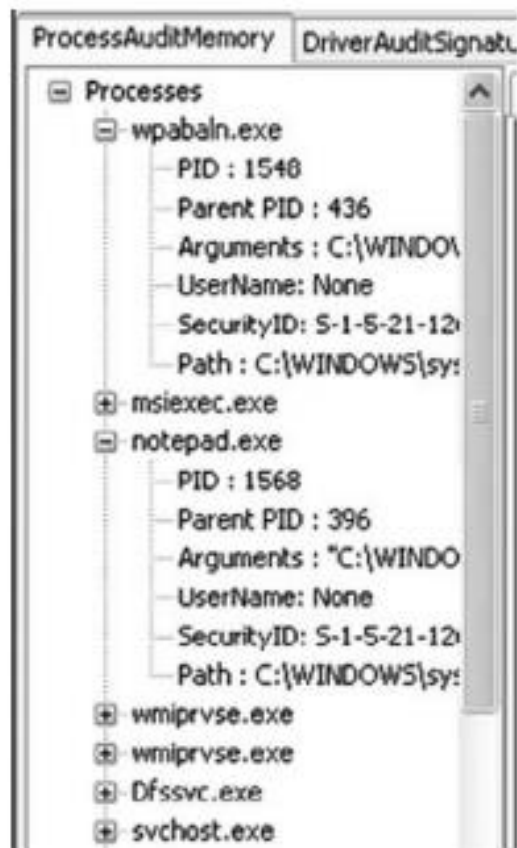
```
Usage: process.bat
-input      name of snapshot. Exclude for live memory.
-pid        PID of the process to inspect. Default: 4294967295 = All
-process    optional name of the process to inspect. Default: Excluded
-handles    true|false inspect all the process handles. Default: false
-sections   true|false inspect all process memory ranges. Default: false
-ports      true|false inspect all the ports of a process. Default: false
-strings    true|false inspect all the strings of a process. Default: false
-output     directory to write the results. Default .\Audits
```

Theo mặc định, dòng lệnh trước đặt các tệp XML kết quả của nó vào chỉ mục. \Audit . Trong trường hợp này, đường dẫn đầy đủ là D:\mandiant\Audits\WINTERMUTE\ 20090103134554 trình xem Audit Mandiant là một công cụ GUI cung cấp cho nhà phân tích giao diện đồ họa vào các tệp XML được tạo bằng cách sử dụng Memoryze để phân tích các kết xuất bộ nhớ. Để khởi động Audit Viewer, bấm đúp vào tệp *AuditViewer.py* trong thư mục mà bạn đã giải nén lưu trữ được tải xuống từ trang Mandiant. Khi bạn cài đặt các mô-đun *wxPython*, bạn sẽ thấy GUI *Audit Viewer* mở, tại thời điểm đó bạn sẽ cấu hình công cụ bằng cách thay đổi thư mục cài đặt bộ nhớ (nếu cần), chọn chạy trên hộp kiểm hình ảnh và cung cấp đường dẫn đến tệp hình ảnh



Hình 11: GUI Audit Viewer hiển thị thay đổi cấu hình

Khi bạn đã thực hiện các thay đổi cần thiết, hãy nhấp vào nút mở ở đầu giao diện người dùng Audit Viewer và điều hướng đến thư mục chứa tệp XML đã được tạo ra. Khi thư mục đã được chọn, trình xem kiểm toán sẽ phân tích các tệp có sẵn và điền vào cây Processes trong giao diện người dùng, như trong Hình 12.



Hình 12: Process tree trong GUI Audit Viewer

Hình 12 cũng minh họa hai quá trình được mở rộng để hiển thị các đối số PID, PPID (hoặc dòng lệnh), cũng như các thông tin khác về mỗi quy trình. Để đào sâu hơn vào từng xử lý, bấm đúp vào tên quy trình trong cây quy trình, rồi xem nội dung của các tab khác nhau (Tập, Thư mục, v.v.) hiển thị trong giao diện người dùng Audit Viewer, như Hình 13 minh họa.



*Hình 13: Tab hiển thị chi tiết tiến trình*

Memoryze và Audit Viewer cung cấp một số tùy chọn mở rộng cho nhà phân tích. Ví dụ: dựa trên những phát hiện trong Audit Viewer, bạn có thể có được một hình ảnh của một quá trình thực thi từ tệp hình ảnh. Để làm như vậy, sử dụng processdd.bat tập tin bó như sau:

D:\mandiant\processdd.bat -pid PID -input d:\hacking\boomer-win2003.img

Bạn cũng có thể sử dụng các tệp khác được cung cấp với Memoryze để thực hiện rootkit và phát hiện hook, cũng như tìm kiếm trình điều khiển ([www.mandiant.com/software/usememoryze.html](http://www.mandiant.com/software/usememoryze.html)). Blog Mandiant M-union (<http://blog.mandiant.com/>) cung cấp các ví dụ bổ sung về cách sử dụng Memoryze

và Audit Viewer, chẳng hạn như cách tích hợp hai công cụ vào phần mềm hướng dẫn Ứng dụng phân tích pháp y từ EnCase.

#### *1.4.26. HBGary Responder*

HBGary Responder là một công cụ phân tích kết xuất bộ nhớ GUI thương mại được mô tả trên trang web của công ty ([www.hbgy.com](http://www.hbgy.com)) dưới dạng bộ nhớ trực tiếp và thời gian chạy bộ phần mềm phân tích được sử dụng để phát hiện, chẩn đoán và phản hồi với máy tính tiên tiến hiện nay các mối đe dọa. Cũng như các công cụ phân tích khác, các Responder (phiên bản Professional và Phiên bản Field Editions) cho phép người trả lời phân tích cú pháp và phân tích kết xuất bộ nhớ có thể hạn chế được khả năng bị xâm nhập hoặc nhiễm API hệ thống.. Mặc dù Responder được viết với giống như phân tích phần mềm độc hại, nó cũng là một công cụ nhanh và có khả năng cung cấp rất nhiều chức năng liên quan đến ứng phó sự cố và rất dễ dàng cho người trả lời sử dụng để lấy thông tin họ cần một cách nhanh chóng.

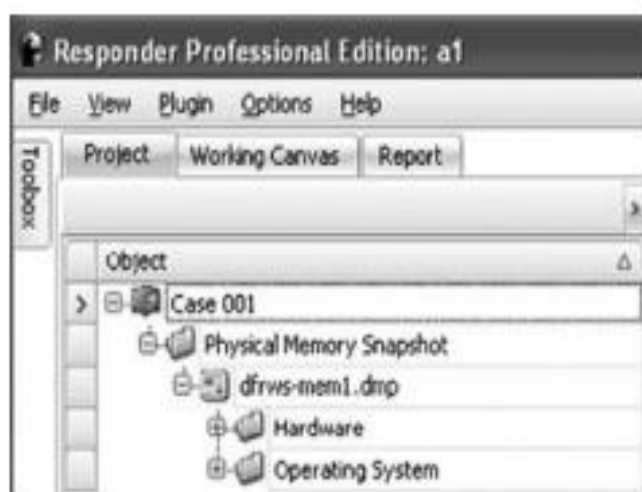
#### **Chú ý:**

Một bản đánh giá của phiên bản Responder Professional phiên bản 1.3.0.377 đã được sử dụng trong các ví dụ được liệt kê trong phần này của chương. Tuy nhiên, chức năng được trình bày và quan sát trong phần này của cả chuyên gia và các sản phẩm Field Edition. Chúng tôi sẽ không tiến hành đánh giá toàn diện về sản phẩm Responder (sản phẩm chuyên nghiệp bao gồm tháo gỡ nhị phân, điều khiển đồ thị dòng chảy và kỹ thuật đảo ngược), làm như vậy là vượt quá phạm vi của cuốn sách này và chúng tôi đang tập trung vào các khía cạnh của sản phẩm liên quan trực tiếp nhất đến phân tích kết xuất bộ nhớ đến các hoạt động ứng phó sự cố. Tất cả những gì bạn cần làm để bắt đầu phân tích kết xuất

bộ nhớ với Responder Professional là tạo một trường hợp và sau đó nhập snapshot bộ nhớ vật lý bằng cách chọn tệp từ thanh menu, sau đó nhập và sau đó nhập snapshot bộ nhớ vật lý. Ví dụ, chúng tôi sẽ sử dụng kết xuất bộ nhớ Windows 2000 đầu tiên từ bộ nhớ DFRWS 2005; tuy nhiên, giống như Memoryze, Responder hoạt động với các bộ nhớ từ Windows 2000, tất cả cách lên thông qua các phiên bản mới nhất của Windows.

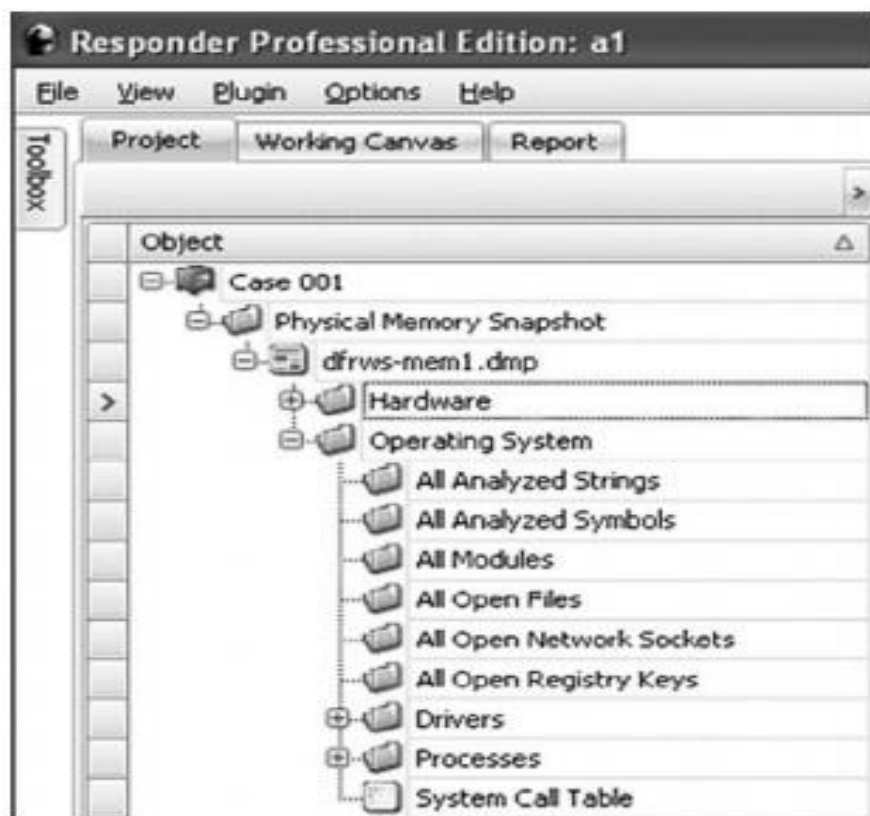
Khi nhập một bộ nhớ kết xuất tập tin snapshot vào trường phản hồi, một tùy chọn có sẵn để trích xuất và phân tích tất cả các bit đáng ngờ. Các quy tắc mà Responder sử dụng để xác định những gì cấu thành nên nghi ngờ của người dùng là dựa trên một tệp văn bản mà bạn có thể mở và xem xét, và thậm chí bạn có thể thêm hoặc xóa nhận xét, tái hiện lại các lỗi, bạn cũng có thể thêm các mục vào tệp dựa trên kinh nghiệm, từ đó tăng thêm hiệu quả.

Khi tệp kết xuất bộ nhớ đã được nhập và phân tích cú pháp, Responder sẽ hiển thị trong ngăn bên trái tệp kết xuất bộ nhớ với hai thư mục, phần cứng và hệ điều hành, như Hình 14 minh họa.



Hình 14: Tệp kết xuất bộ nhớ

Mở rộng thư mục bên dưới thư mục phần cứng sẽ hiển thị bảng ngắt. Mở rộng thư mục hệ điều hành sẽ hiển thị rất nhiều thông tin bổ sung, bao gồm các quy trình và tất cả các ổ cắm mạng mở (một phần, những gì người phản hồi có thể là nhất quan tâm đến), như trong Hình 15.



Hình 15: Thư mục hệ điều hành được mở rộng trong GUI

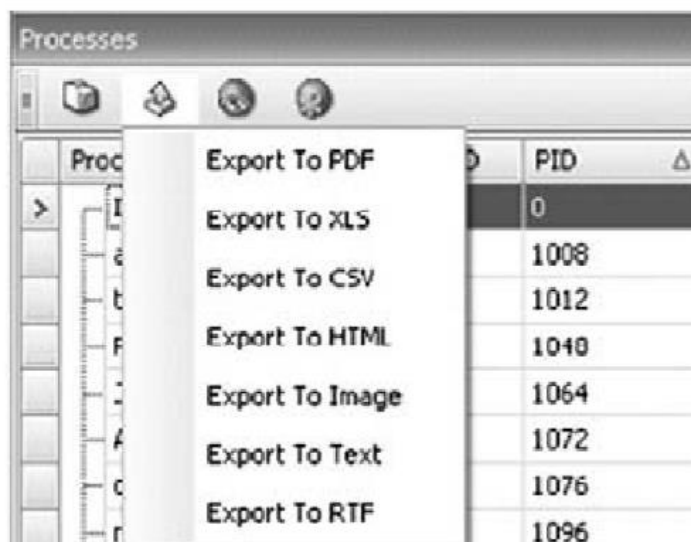
Sau đó, bạn có thể mở rộng thư mục quy trình để xem tất cả các quy trình hoạt động được trích xuất từ kết xuất bộ nhớ hoặc bấm đúp vào thư mục Processes để có các quy trình và chi tiết thông tin về từng quy trình hiển thị trong framework bên phải giao diện Responder, như trong Hình 16.



nc.exe	592	1096	"c:\winnt\system32\nc.exe" -L -p 3000 -t -e cmd.exe
cmd2k.exe	324	1112	"E:\Shells\cmd2k.exe" /D /T:80 /F:ON /K cmdenv.bat
cmd2k.exe	324	1132	"E:\Shells\cmd2k.exe" /D /T:80 /F:ON /K cmdenv.bat
smss.exe	8	156	\SystemRoot\System32\smss.exe
winlogon.exe	156	176	winlogon.exe
csrss.exe	156	180	C:\WINNT\system32\csrss.exe ObjectDirectory=\Windows SharedSection=1024,3072,51...
services.exe	176	228	C:\WINNT\system32\services.exe
lsass.exe	176	240	
dd.exe	1112	284	..\Acquisition\FAU\dd.exe if=\\.\PhysicalMemory of=F:\intrusion2005\physicalmemory.dd ...
helix.exe	820	324	E:\helix.exe

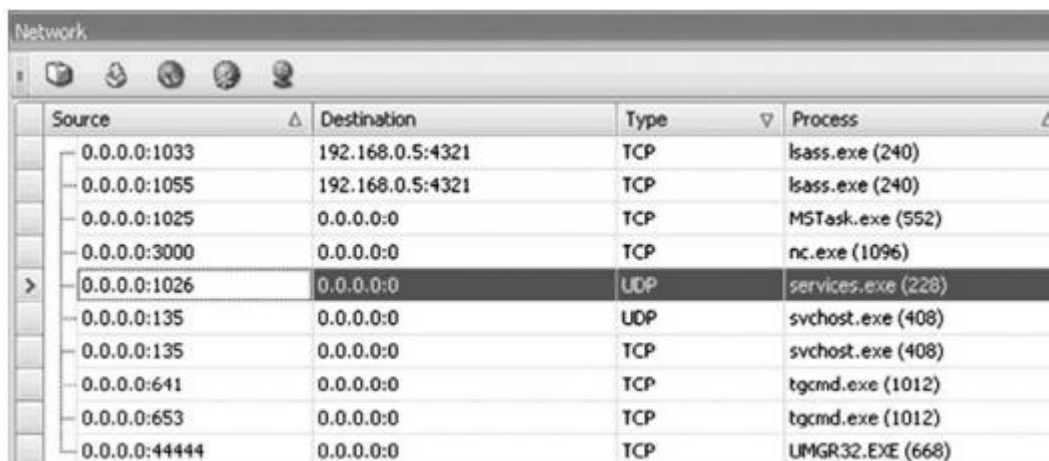
*Hình 16: GUI qui trình chi tiết*

Giao diện của Responder cung cấp nhiều thông tin hữu ích. Bạn cũng có thể điều chỉnh các cột bằng cách chọn chúng và kéo chúng sang cột mới vị trí trong cùng một framework view. Ngoài ra, bạn có thể xuất thông tin từ framework view sang các định dạng khác nhau bằng cách nhấp vào biểu tượng thích hợp ở trên cùng của framework view, như Hình 17 minh họa.



*Hình 17: Option xuất dữ liệu*

Bấm đúp vào thư mục tất cả các ổ cắm mạng mở sẽ mở tab mạng trong framework, tương tự như đầu ra của *netstat.exe*, như trong Hình 18.



Source	Destination	Type	Process
0.0.0.0:1033	192.168.0.5:4321	TCP	lsass.exe (240)
0.0.0.0:1055	192.168.0.5:4321	TCP	lsass.exe (240)
0.0.0.0:1025	0.0.0.0:0	TCP	MSTask.exe (552)
0.0.0.0:3000	0.0.0.0:0	TCP	nc.exe (1096)
0.0.0.0:1026	0.0.0.0:0	UDP	services.exe (228)
0.0.0.0:135	0.0.0.0:0	UDP	svchost.exe (408)
0.0.0.0:135	0.0.0.0:0	TCP	svchost.exe (408)
0.0.0.0:641	0.0.0.0:0	TCP	tgcmd.exe (1012)
0.0.0.0:653	0.0.0.0:0	TCP	tgcmd.exe (1012)
0.0.0.0:44444	0.0.0.0:0	TCP	UMGR32.EXE (668)

Hình 18: Network GUI

Bạn cũng có thể xem các thẻ điều khiển đang mở cho tất cả các quy trình bằng cách nhấp đúp vào tất cả thư mục Open Files, như Hình 19 minh họa.

audit.log	\intrusion2005\audit.log	dd.exe (284)
physicalmemory.dd	\intrusion2005\physicalmemory.dd	dd.exe (284)
shells	\shells	dd.exe (284)
hxdef-rk100sb4d1ba5d	\hxdef-rk100sb4d1ba5d	dfrws2005.exe (592)
hxdef-rk100sb4d1ba5d	\hxdef-rk100sb4d1ba5d	dfrws2005.exe (592)
ntcontrolpipe9	\net\ntcontrolpipe9	dfrws2005.exe (592)
svcctl	\svcctl	dfrws2005.exe (592)

Hình 19: Danh sách các tệp đang mở

Bạn có thể xem các thẻ điều khiển đang mở cho một quy trình cụ thể bằng cách mở

rộng cây cho từng xử lý và chọn Open Files Bạn có thể làm tương tự cho các ổ cắm mạng mở và mở phím đăng ký. Ngoài việc có thể xem nhanh tất cả các thông tin này, cả trên bộ nhớ định dạng kết xuất rộng cũng như trên định dạng theo quy trình, bạn có thể phân tích các hình ảnh thực thi (tập tin .exe và...) cho chuỗi cũng như tìm kiếm các mục cụ thể trong kết xuất bộ nhớ sử dụng các chuỗi con, biểu thức chính quy hoặc kết hợp chính xác. Có khả năng sắp xếp các mục có thể nhìn thấy trong các cột cũng cho phép bạn xác định các quy trình hoặc mô-đun đáng ngờ (ví dụ: DLL) nhanh hơn nữa

#### *1.4.27. Parsing Process Memory*

Trước đây, các nhà phân tích đã sử dụng các công cụ như String.exe hoặc grep tìm kiếm để phân tích nội dung của kết xuất RAM và tìm kiếm các chuỗi (mật khẩu), địa chỉ IP hoặc e-mail, URL và những thứ tương tự. Tuy nhiên, khi bạn phân tích cú pháp thông qua một tệp có kích thước khoảng nửa megabyte, thì đó là một trường hợp phức tạp hơn với bạn tìm thấy. Đôi khi một phân tích viên sẽ mở vài tập tin trong trình soạn thảo hex và xác định chuỗi đặc biệt, và nếu thấy những gì giống như là tên người dùng, cô ấy có thể cho rằng chuỗi là mật khẩu. Tuy nhiên, phân tích một tệp kết xuất RAM theo cách này không cho phép phân tích tương quan chuỗi đó với một quá trình cụ thể. Ghi nhớ ví dụ về Nguyên tắc trao đổi Locard từ Chương 1. Nếu chúng ta thu thập nội dung của bộ nhớ vật lý trong ví dụ này, chúng ta không thể nói rằng tồn tại một địa chỉ IP cụ thể hoặc dữ liệu khác, chẳng hạn như danh sách thư mục, đã được gắn với một sự kiện hoặc quá trình cụ thể. Tuy nhiên, nếu chúng tôi sử dụng thông tin được cung cấp trong cấu trúc quy trình trong bộ nhớ và định vị tất cả các trang mà quy trình sử dụng vẫn còn trong bộ nhớ khi nội dung bị đổ, sau đó chúng tôi có thể chạy các tìm kiếm và xác định quá trình đã sử dụng thông tin đó.

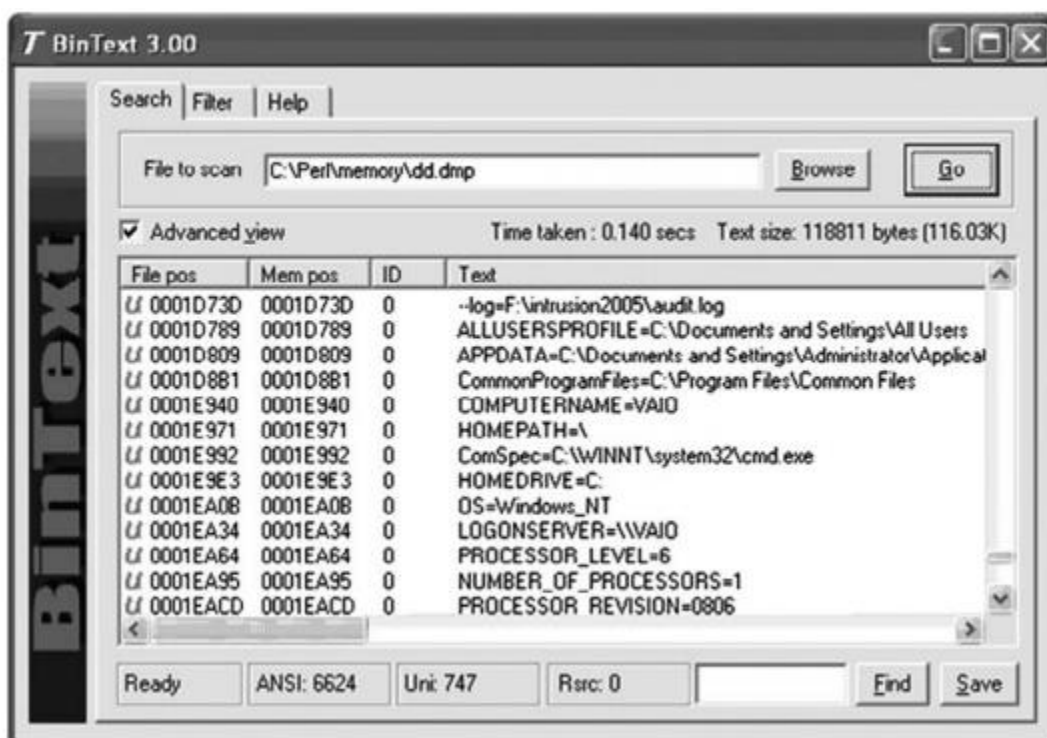
Công cụ *lspm.pl* cho phép bạn thực hiện việc này một cách tự động khi làm việc với Windows 2000 memory dumps. *Lspm.pl* có cùng các đối số như *lspd.pl* (tên, đường dẫn của kết xuất tệp và phần bù vật lý trong tệp của cấu trúc quy trình) trích xuất phần có sẵn các trang từ tệp kết xuất, ghi chúng vào một tệp trong thư mục làm việc hiện tại. Chạy *lspm.pl* chống lại quá trình *dd.exe*, sử dụng dòng lệnh sau:

```
C:\perl\memory>lspm.pl d:\dumps\dfrrws1-mem.dmp 0x0414dd60
```

Đầu ra như sau:

```
Name : dd.exe -> 0x01d9e000
There are 372 pages (1523712 bytes) to process.
Dumping process memory to dd.dmp...
Done.
```

Bây giờ bạn có một tệp có tên *dd.dmp* có kích thước 1,523,712 byte và chứa tất cả các trang bộ nhớ (tổng cộng 372) cho quy trình đó vẫn còn khả dụng khi tạo tệp kết xuất. Bạn có thể chạy *String.exe* hoặc sử dụng *BinText* (được minh họa trong Hình 20) từ Foundstone.com để phân tích thông qua tệp tìm kiếm chuỗi Unicode và ASCII hoặc chạy các tìm kiếm grep cho địa chỉ IP hoặc e-mail và thẻ tín dụng hoặc số An sinh xã hội.



Hình 20: Nội dung của bộ nhớ tiến trình trong BinText

Trong Hình 20, bạn có thể thấy một số chuỗi Unicode có trong bộ nhớ được sử dụng theo quy trình *dd.exe*, bao gồm tên của hệ thống và tên của LogonServer

; một khía cạnh quan trọng của trường hợp này là bạn có thể có cái nhìn tương quan về những gì được tìm thấy trong một quá trình cụ thể.

Tính không ổn định có tính năng tương tự trong lệnh *memdump*. Như đã đề cập trước đây trong chương này, bạn có thể sử dụng lệnh *memdump* Volatility để kết xuất địa chỉ bộ nhớ cho một quá trình từ kết xuất bộ nhớ Windows XP, như sau:

***D:\Volatility>python volatility memdump -f d:\hacking\xp-laptop1.img -p 4012***

**Tip:**

Bạn có thể sử dụng *Volatility* để thu thập bộ nhớ quy trình cho các quy trình bị ẩn bởi rootkit, ngay cả những ẩn được sử dụng thao tác đối tượng kernel trực tiếp

(DKOM) kỹ thuật (xem Chương 7). Cụ thể, kỹ thuật DKOM không liên kết được khối EProcess cho quy trình ẩn từ danh sách quy trình hoạt động được liên kết đôi mà hệ điều hành mà Google nhìn thấy. Tuy nhiên, bằng cách sử dụng *Volatility* để kiểm tra tệp kết xuất bộ nhớ thô hoặc tệp đóng băng của Windows XP, bạn có thể tìm kiếm các quy trình không phải là một phần của danh sách liên kết đôi đó (sẽ thảo luận sau trong phần chương), và sau đó sử dụng lệnh *memdump* để lấy bộ nhớ đã sử dụng bởi quá trình từ tệp tin kết xuất bộ nhớ.

#### 1.4.28. *Extracting the Process Image*

Như bạn đã thấy trước đó trong chương này, khi một quy trình được khởi chạy, tệp thực thi được đọc vào bộ nhớ. Một trong những thông tin mà bạn có thể nhận được từ các chi tiết của quy trình (thông qua *lspd.pl*) là phần bù trong tệp kết xuất bộ nhớ Windows 2000 đến Địa chỉ hình ảnh. Như bạn đã thấy, *lspd.pl* sẽ kiểm tra nhanh để xem có thể tìm thấy hình ảnh thực thi không tại vị trí đó. Một trong những điều bạn có thể làm để phát triển thông tin này hơn nữa là phân tích cú pháp tiêu đề tệp PE (nội dung chúng tôi sẽ đề cập chi tiết trong Chương 6) và xem cho dù bạn có thể trích xuất toàn bộ nội dung của hình ảnh thực thi từ tệp tin kết xuất bộ nhớ Windows 2000. *Lspi.pl* cho phép bạn làm điều này tự động. *Lspi.pl* là một tập lệnh Perl có cùng các đối số như *lspd.pl* và *lspm.pl* định vị sự bắt đầu của hình ảnh thực thi cho quá trình đó. Nếu Base Image offset không thực sự dẫn đến một tệp hình ảnh thực thi, *lspi.pl* sẽ phân tích các giá trị có trong tiêu đề PE để xác định vị trí các trang tạo nên phần còn lại của tệp hình ảnh thực thi. Vì vậy bạn có thể chạy *lspi.pl* theo quy trình *dd.exe* (với PID là 284) bằng cách sử dụng dòng lệnh sau:

*c:\perl\memory>lspi.pl d:\dumps\dfrrs1-mem.dmp 0x0414dd60*

Đầu ra của lệnh xuất hiện như sau:

```
Process Name      : dd.exe
PID               : 284
DTB               : 0x01d9e000
PEB               : 0x7ffdf000 (0x02c2d000)
ImgBaseAddr       : 0x00400000 (0x00fee000)
e_lfanew = 0xe8
NT Header = 0x4550
Reading the Image File Header
Sections = 4
Opt Header Size = 0x000000e0 (224 bytes)
Characteristics:
    IMAGE_FILE_EXECUTABLE_IMAGE
    IMAGE_FILE_LOCAL_SYMS_STRIPPED
    IMAGE_FILE_RELOCS_STRIPPED
    IMAGE_FILE_LINE_NUMS_STRIPPED
    IMAGE_FILE_32BIT_MACHINE
Machine = IMAGE_FILE_MACHINE_I860
Reading the Image Optional Header
Opt Header Magic = 0x10b
Subsystem         : IMAGE_SUBSYSTEM_WINDOWS_CUI
Entry Pt Addr     : 0x00006bda
Image Base        : 0x00400000
File Align        : 0x00001000
Reading the Image Data Directory information
Data Directory     RVA          Size
-----
ResourceTable     0x0000d000    0x00000430
DebugTable        0x00000000    0x00000000
BaseRelocTable    0x00000000    0x00000000
DelayImportDesc   0x0000af7c     0x000000a0
TLSTable          0x00000000    0x00000000
GlobalPtrReg      0x00000000    0x00000000
ArchSpecific      0x00000000    0x00000000
CLIHeader         0x00000000    0x00000000
LoadConfigTable   0x00000000    0x00000000
ExceptionTable    0x00000000    0x00000000
ImportTable       0x0000b25c    0x000000a0
unused            0x00000000    0x00000000
```

```

BoundImportTable      0x00000000      0x00000000
ExportTable           0x00000000      0x00000000
CertificateTable      0x00000000      0x00000000
IAT                   0x00007000      0x00000210
Reading Image Section Header Information
Name      Virt Sz      Virt Addr      rData Ofc      rData Sz      Char
-----
.text     0x00005ee0      0x00001000      0x00001000      0x00006000      0x60000020
.data     0x000002fc      0x0000c000      0x0000c000      0x00001000      0xc0000040
.rsrc     0x00000430      0x0000d000      0x0000d000      0x00001000      0x40000040
.rdata    0x00004cfa      0x00007000      0x00007000      0x00005000      0x40000040
Reassembling image file into dd.exe.img
Bytes written = 57344
New file size = 57344

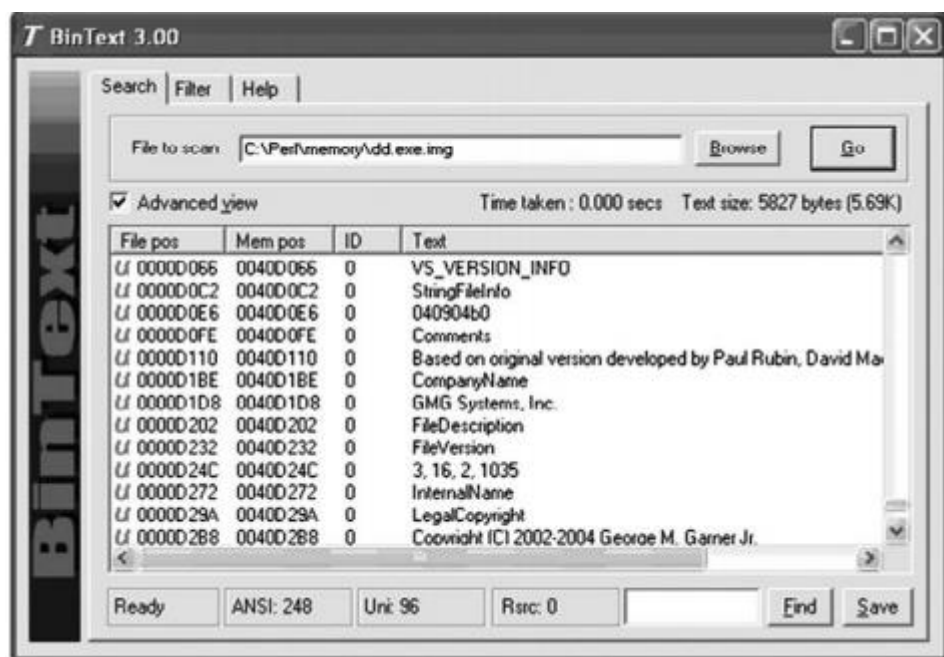
```

Như bạn có thể thấy, đầu ra của *lspl.pl* khá dài dòng và nhiều thông tin được hiển thị có thể không hữu ích đối với (hoặc được hiểu bởi) một phân tích viên trừ khi phân tích viên đó quan tâm đến phân tích phần mềm độc hại. Một lần nữa, chúng ta sẽ thảo luận chi tiết về thông tin này trong Chương 6. Hiện tại, các yếu tố quan là các thông tin trong phần ““Reading Image Section Header Information”. “Reading Image Section Header Information” cung cấp cho bạn bản đồ đường đi để gắn lại tệp thực thi hình ảnh vì nó cho phép bạn biết nơi tìm các trang tạo nên tệp hình ảnh đó. *Lspi.pl* sử dụng bản đồ này và cố gắng ghép lại hình ảnh thực thi thành một tệp. Nếu nó thành công, nó sẽ ghi ra tệp tin dựa trên tên của quá trình, với .img được nối thêm (để ngăn chặn việc thực hiện ngẫu nhiên của tệp tin). *Lspi.pl* sẽ không tập hợp lại tệp tin nếu có bất kỳ các trang bộ nhớ đã được đánh dấu là không hợp lệ và không còn nằm trong bộ nhớ (ví dụ: chúng đã được phân trang ra tệp hoán đổi, pagefile.sys). Thay vào đó, *lspl.pl* sẽ báo cáo rằng nó không thể tập hợp lại tệp tin hoàn chỉnh vì một số trang (thậm chí chỉ một) không có sẵn trong bộ nhớ.

Bây giờ, tệp bạn trích xuất từ kết xuất bộ nhớ sẽ không hoàn toàn giống với tệp tin thực thi gốc. Điều này là do một số phần của tệp tin có thể ghi được và những phần đó các phần sẽ thay đổi khi quá trình thực hiện. Trong quá trình thực thi, các yếu tố khác nhau của mã thực thi (địa chỉ, biến, v.v.) sẽ thay đổi theo môi



trường và giai đoạn thực hiện. Tuy nhiên, có một số cách bạn có thể xác định bản chất của một tập tin và nhận được một số thông tin về mục đích của nó. Một trong những cách đó là để xem liệu tệp có bất kỳ thông tin phiên bản tệp nào được biên dịch vào nó, như được thực hiện với hầu hết các tệp được tạo bởi công ty phần mềm hợp pháp. Như bạn đã thấy từ “Reading Image Section Header Information” ở đó là một phần có tên *.rsrc*, là tên thường được sử dụng cho phần tài nguyên của tệp PE. Phần này có thể chứa nhiều loại tài nguyên, chẳng hạn như hộp thoại và chuỗi phiên bản và tổ chức như một hệ thống các loại tập tin. Sử dụng BinText, bạn có thể tìm chuỗi Unicode VS\_VERSION\_INFO và xem liệu có bất kỳ thông tin nhận dạng nào có sẵn trong tệp tin hình ảnh thực thi. Hình 21 minh họa một số chuỗi được tìm thấy trong tệp *dd.exe.img* sử dụng BinText



Hình 21: Chuỗi phiên bản được tìm thấy trong *dd.exe.img* với BinText

Một phương pháp khác để xác định bản chất của tệp là sử dụng băm tập tin. Bạn vừa nói tệp tin được tạo bởi *lspi.pl* isn chính xác giống như tệp tin gốc, vậy làm thế nào chúng ta có thể sử dụng băm?, Vâng, bạn có thể đúng với một điểm.

Chúng ta có thể sử dụng băm MD5 để so sánh, vì như chúng ta biết, việc thay đổi chỉ 1 bit sẽ tính toán ra một hàm băm hoàn toàn khác. Vậy chúng ta có thể làm gì?

Vào mùa hè năm 2006, Jesse Kornblum đã phát hành một công cụ có tên là *ssdeep* (<http://ssdeep.sourceforge.net>) thực hiện một cái gì đó gọi là băm piecewise kích hoạt theo ngưỡng, hoặc băm mờ. Để hiểu chi tiết về những gì điều này đòi hỏi, hãy chắc chắn đọc bài viết của Jesse, DFRWS 2006 về chủ đề này. Tóm lại,

Jesse đã triển khai một thuật toán sẽ cho bạn biết tỷ lệ phần trăm giống hệt nhau

chuỗi các bit mà các tệp có điểm chung, dựa trên giá trị băm của chúng và được tính bằng *ssdeep*. Bởi vì chúng tôi biết rằng trong trường hợp này, phiên bản *dd.exe* của George Garner đã được sử dụng để kết xuất nội dung của RAM từ hệ thống Windows 2000 cho DFRWS 2005 Memory Challenge, chúng ta có thể so sánh tệp *dd.exe.img* với tệp *dd.exe* gốc mà chúng ta vừa có sẵn.

Đầu tiên, chúng tôi bắt đầu bằng cách sử dụng *ssdeep.exe* để tính toán hàm băm cho tệp hình ảnh:

```
D:\tools>ssdeep c:\perl\memory\dd.exe.img > dd.sdp\
```

Bây giờ chúng tôi đã tạo ra hàm băm và lưu thông tin vào tệp *dd.sdp*. Sử dụng

các công tắc khác có sẵn cho *ssdeep.exe*, chúng tôi có thể nhanh chóng so sánh tệp *.img* với bản gốc hình ảnh thực thi:

```
D:\tools>ssdeep -v -m dd.sdp d:\tools\dd\old\dd.exe d:\tools\dd\old\dd.exe  
matches c:\perl\memory\dd.exe.img (97)
```

Chúng ta cũng có thể thực hiện việc này trong một dòng lệnh bằng cách sử dụng khóa chuyển đổi hoặc chuyển đổi:

```
D:\tools|> ssdeep -d c:\perl\memory\dd.exe.img d:\tools\dd\old\dd.exe
C:\perl\memory\dd.exe.img matches d:\tools\dd\old\dd.exe (97)
```

Chúng ta thấy rằng tệp hình ảnh được tạo bởi lspi.pl có khả năng khớp 97 phần trăm tệp dd.exe gốc.

Hãy nhớ rằng, để so sánh băm hoạt động chính xác, chúng ta cần một cái gì đó mà chúng ta có thể so sánh các tệp được tạo bởi lspi.pl. Ssdeep.exe là một công cụ tương đối mới, mặc dù cực kỳ mạnh mẽ, công cụ và có thể sẽ mất thời gian đợi khi các bộ băm được tạo bằng ssdeep.exe hoặc kết hợp băm được tính bằng ssdeep.exe.

Chúng ta có thể sử dụng Volatirity Framwork để cố gắng trích xuất hình ảnh thực thi từ một tập tin kết xuất bộ nhớ Windows XP bằng lệnh *procdump*. Cú pháp lệnh procdump(từ tệp readme.txt) xuất hiện như sau:

```
procdump
-----
For each process in the given image, extract an executable sample.
If -t and -b are not specified, Volatility will attempt to infer
reasonable values.
Options:
-f      <Image>    Image file to load
-b      <base>     Hexadecimal physical offset of valid Directory Table Base
-t      <type>     Image type (pae, nopae, auto)
-o      <offset>   Hexadecimal physical offset of EPROCESS object
-p      <pid>      Pid of process
-m      <mode>     Strategy to use when extracting executable sample.
                  Use "disk" to save using disk-based section sizes or "mem"
                  for memory based sections (default": "mem").
```

Tiếp tục với ví dụ Volatility từ trước đó trong chương này, chúng ta có thể trích xuất tệp hình ảnh thực thi cho quá trình dd.exe (PID 4012 trong bộ nhớ xp laptop1.img kết xuất tập tin) bằng lệnh này:

```
D:\Volatility>python volatility procdump -f d:\hacking\xp-laptop1.img -p
4012
```

```
Dumping dd.exe, pid: 4012 output: executable.4012.exe
```

```
D:\Volatility>dir exe*.exe
```

***Volume in drive D is Data***

***Volume Serial Number is 8049-F885***

***Directory of D:\Volatility***

***01/01/2009 12:45 PM 57,344 executable.4012.exe***

***1 File(s) 57,344 bytes***

Mặc dù không dài dòng như tập lệnh *Perl lspil.pl* cho bộ nhớ Windows 2000, lệnh *procdump Volatility* trích xuất tệp hình ảnh thực thi cho quá trình. Điều này có thể cực kỳ hữu ích trong quá trình phân tích phần mềm độc hại, vì rất nhiều phần mềm độc hại hiện tại bị xáo trộn (được mã hóa, nén hoặc cả hai) trên đĩa, khiến việc phân tích tĩnh (xem Chương 6) trở nên khó khăn. Ngoài ra, một số phần mềm độc hại có thể chỉ tồn tại trong bộ nhớ, không bao giờ thực sự được ghi vào hard drive; có thể trích xuất tệp hình ảnh thực thi từ kết xuất bộ nhớ là cách duy nhất để có được một bản sao của tập tin để phân tích.

### **Chú ý:**

Responder thường có thể phải đối mặt với các hệ thống sử dụng một số loại mã hóa của một khối lượng cụ thể hoặc toàn bộ đĩa. Tôi đã gặp trường hợp này và khi tôi phải tiến hành mua lại với ý định thực hiện phân tích (trái ngược với việc đơn giản là có được một hình ảnh), tôi đã chọn để thực hiện việc mua lại ổ cứng. Năm 2007, Brian Kaplan đã viết một bài luận văn có tựa đề *RAM dump in isolation that is*: Trích xuất đĩa khóa mã hóa từ bộ nhớ dễ bay hơi. Cùng với bài viết đó, Brian cũng đã phát hành một bằng chứng về công cụ để trích xuất Pretty Good Privacy (PGP) các khóa mã hóa toàn bộ đĩa (WDE) từ kết xuất bộ nhớ. Tài liệu và bằng chứng về công cụ khái niệm có sẵn từ [www.andrew.cmu.edu/user/bfkaplan/](http://www.andrew.cmu.edu/user/bfkaplan/) / # KeyExtraction

#### 1.4.29. *Memory Dump Analysis and the Page File*

Cho đến nay, chúng tôi đã xem xét phân tích cú pháp và phân tích nội dung của một bãi chứa RAM trong sự cô lập đó là, không sử dụng các thông tin bổ sung. Điều này có nghĩa là các công cụ như *lspm.pl* chỉ dựa vào nội dung của kết xuất RAM sẽ cung cấp bộ nhớ không đầy đủ kết xuất, vì các trang bộ nhớ đã được hoán đổi thành tệp trang (*pagefile.sys* trên các hệ thống Windows) sẽ không được tích hợp vào kết xuất bộ nhớ kết quả. Vượt qua sự thiếu sót này, vào mùa xuân năm 2006 Nicholas Paul Maclean đã xuất bản luận án của mình, mua lại và phân tích bộ nhớ Windows, giải thích các hoạt động bên trong của hệ thống quản lý bộ nhớ Windows và cung cấp một công cụ nguồn mở gọi là *vtop* (được viết bằng Python) để xây dựng lại không gian địa chỉ ảo của một quá trình.

Đầu năm 2007, bài báo của Jesse Kornblum xuất bản trên tạp chí kỹ thuật số phân tích và nhà xuất bản tạp chí cho phép Jesse đăng một bản sao của bài báo này lên trang mạng.

Trong bài báo này, Jesse thể hiện các khía cạnh của dịch địa chỉ trang và cách tập tin trang có thể được kết hợp vào quá trình phân tích bộ nhớ để thiết lập một bản hoàn chỉnh hơn (và chính xác) xem các thông tin có sẵn.

#### 1.4.30. *Pool Allocations*

Khi trình quản lý bộ nhớ Windows phân bổ bộ nhớ, chia 4 KB(4096 byte) mỗi trang. Tuy nhiên, phân bổ toàn bộ trang 4 KB, giả sử, một câu được sao chép vào Clipboard sẽ là một sự lãng phí bộ nhớ. Vì vậy, trình quản lý bộ nhớ phân bổ một số trang trước thời hạn, giữ một bộ nhớ có sẵn. Andreas Schuster đã thực hiện nghiên cứu trong lĩnh vực này và mặc dù Microsoft cung cấp danh sách các tiêu đề nhóm được sử dụng để chỉ định các nhóm thường được sử dụng. Nhiều Pool headers thường được sử dụng được liệt kê trong *pooltag*. tệp *txt*

([www.microsoft.com/whdc/do/tips/PoolMem.msp](http://www.microsoft.com/whdc/do/tips/PoolMem.msp)) được cung cấp với Microsoft Debugging Tools và Microsoft cung cấp bài viết Cơ sở tri thức mô tả cách định vị thẻ / tiêu đề nhóm được sử dụng bởi các ứng dụng của bên thứ ba (<http://support.microsoft.com/default.aspx?scid=kb;en-us;298102>). Andreas đã sử dụng một phương pháp tương tự để xác định định dạng nhóm bộ nhớ được sử dụng để lưu giữ thông tin về các kết nối mạng trong kết xuất bộ nhớ Windows 2000

([http://computer.forensikblog.de/en/2006/07/finding\\_network\\_socket\\_Activity\\_in\\_pools.html](http://computer.forensikblog.de/en/2006/07/finding_network_socket_Activity_in_pools.html)) ông đã tìm kiếm Pool header trong trình điều khiển *tcpip.sys* trên Windows 2000 và có thể xác định định dạng của thông tin kết nối mạng trong nhóm bộ nhớ.

Nhược điểm của việc tìm kiếm phân bổ nhóm bộ nhớ là mặc dù các tiêu đề nhóm

dường như không thay đổi giữa các phiên bản Windows, định dạng của dữ liệu cư trú trong nhóm bộ nhớ thay đổi và không có tài liệu nào có sẵn về định dạng của những nhóm bộ nhớ.

## **Tóm lược**

Bây giờ, rõ ràng là bạn có một số tùy chọn để thu thập vật lý hoặc quy trình bộ nhớ từ một hệ thống trong quá trình ứng phó sự cố. Trong Chương 1, chúng tôi đã kiểm tra một số các công cụ để thu thập các phần khác nhau của bộ nhớ dễ bay hơi trong quá trình phản hồi trực tiếp (các quy trình, kết nối mạng và tương tự), hãy nhớ rằng luôn có khả năng tiềm ẩn cho API Windows (dựa trên công cụ) bị kẻ tấn công xâm phạm. Điều này đúng trong bất kỳ trường hợp nào mà phản hồi trực tiếp đang được thực hiện và do đó chúng tôi có thể quyết định sử dụng nhiều phương tiện khác nhau để thu thập thông tin dễ bay hơi. Một rootkit có thể che giấu sự tồn tại của một quy trình từ hầu hết các công cụ liệt kê danh sách các quy trình đang hoạt động (*tlist.exe*, *pslist.exe*), nhưng việc bỏ nội dung của RAM sẽ

cho phép phân tích viên liệt kê hoạt động và thoát các quy trình cũng như các quy trình được ẩn bằng cách sử dụng rootkit chế độ kernel (tìm hiểu thêm về rootkit trong Chương 7).

#### *1.4.31. Solutions Fast Track*

##### **Collecting Process Memory Thu thập bộ nhớ tiến trình**

Một phản hồi có thể được đưa ra với một tình huống không cần thiết phải thu thập toàn bộ nội dung của bộ nhớ vật lý; thay vào đó, nội dung của bộ nhớ được sử dụng bởi quá trình duy nhất sẽ là đủ.

Các quy trình được ẩn thông qua một số phương tiện (xem Chương 7) có thể không thể nhìn thấy và phân tích viên Thu thập nội dung bộ nhớ của một tiến trình là một tùy chọn chỉ khả dụng cho các quy trình được nhìn thấy trong danh sách quy trình hoạt động của cả hệ điều hành và các phân tích viên tiện ích. sẽ không thể cung cấp định danh quy trình đến các công cụ anh ta đang sử dụng để thu thập bộ nhớ được sử dụng bởi quy trình.

Kết xuất bộ nhớ tiến trình cho phép người phân tích thu thập không chỉ bộ nhớ được sử dụng bởi quá trình có thể tìm thấy trong RAM, nhưng cũng là bộ nhớ nằm trong tập tin trang.

Khi bộ nhớ quy trình được thu thập, thông tin bổ sung về quy trình, chẳng hạn như tay cầm mở và các mô-đun được tải, sau đó có thể được thu thập.

##### **Dumping Physical Memory Kết xuất bộ nhớ vật lý**

Một số phương pháp có sẵn để loại bỏ nội dung của bộ nhớ vật lý. Responder nên biết về các lựa chọn có sẵn cũng như ưu và nhược điểm của để đưa ra một lựa chọn thông minh về phương pháp nào nên được dùng.

Việc bỏ nội dung của bộ nhớ vật lý từ hệ thống trực tiếp có thể gây ra sự cố với tính nhất quán vì hệ thống vẫn hoạt động và xử lý thông tin trong khi kết xuất bộ nhớ đang được tạo.

Khi bỏ nội dung của bộ nhớ vật lý, cả bộ phản hồi và nhà phân tích phải ghi nhớ Nguyên tắc trao đổi Locard.

### Analyzing a Physical Memory Dump Phân tích kết xuất bộ nhớ vật lý

Tùy thuộc vào phương tiện được sử dụng để thu thập nội dung của bộ nhớ vật lý, khác nhau các công cụ có sẵn để trích xuất thông tin hữu ích từ kết xuất bộ nhớ. Công dụng của chuỗi.exe, BinText và grep với các biểu thức chính quy khác nhau đã được phổ biến, và nghiên cứu được thực hiện bắt đầu vào mùa xuân năm 2005 cho thấy cách trích xuất cụ thể các quá trình.

Các bộ nhớ vật lý chứa thông tin và các đối tượng hữu ích như các tiến trình, nội dung của Clipboard và kết nối mạng.

Tiếp tục nghiên cứu trong lĩnh vực này đã chứng minh làm thế nào tập tin trang có thể được sử dụng trong kết hợp với kết xuất RAM để phát triển một bộ thông tin đầy đủ hơn.

### **Các câu hỏi thường gặp**

*Hỏi:* Tại sao tôi nên xóa nội dung RAM từ hệ thống trực tiếp? Lợi ích của việc này?

*Trả lời:* Như chúng ta đã thảo luận trong Chương 1, một lượng thông tin đáng kể có sẵn trên mạng hệ thống có thể cực kỳ quan trọng đối với một cuộc phân tích. Thông tin dễ bay hơi này tồn tại trong bộ nhớ, hoặc RAM, trong khi hệ thống đang chạy. Chúng tôi có thể sử dụng các công cụ của bên thứ ba khác nhau (thảo luận trong Chương 1) để thu thập thông tin này, nhưng điều quan trọng là phải thu thập toàn bộ nội dung của bộ nhớ để chúng tôi không chỉ có một bản ghi đầy đủ thông tin có sẵn, mà còn có thể nhìn thấy những thứ mà có thể không thể nhìn thấy được qua các phương tiện truyền thống (ví dụ: những thứ được ẩn bởi rootkit; xem Chương 7 để biết thêm thông tin về rootkit). Bạn cũng có thể tìm thấy thông tin



liên quan đến các quy trình đã thoát cũng như các tàn dư của quy trình còn lại sau khi hệ thống được khởi động lại.

*Hỏi:* Khi tôi đã bỏ các nội dung của RAM, sau đó tôi có thể làm gì để phân tích chúng?

*Trả lời:* Các nhà phân tích đã từng sử dụng các công cụ tìm kiếm dựa trên tệp tiêu chuẩn để phân tích RAM. Các tìm kiếm String.exe và grep đã được sử dụng để định vị mật khẩu, địa chỉ email, URL và tương tự trong RAM. Hiện tại các công cụ tồn tại để phân tích các bãi RAM cho các quy trình, xử lý chi tiết (dòng lệnh, tay cầm), luồng và các đối tượng khác cũng như trích xuất hình ảnh thực thi, điều này cực kỳ có lợi cho phân tích phần mềm độc hại (xem Chương 6 để biết thêm thông tin về chủ đề này) cũng như các cuộc kiểm tra pháp y máy tính truyền thống hơn.

*Hỏi:* Khi tôi kiểm tra 1 máy tính, tôi nhận thấy có một cửa sổ tin nhắn tức thời (IM) được mở lên. Khi tôi kéo thả lại cửa sổ này để xem nội dung cuộc trò chuyện, nội dung này đã bị xóa, liệu rằng những nội dung này có được truyền trong tiến trình này?

*Trả lời:* Nếu vấn đề mà bạn gặp phải chủ yếu là vấn đề xoay quanh một quá trình có thể nhìn thấy được, đồ toàn bộ nội dung của bộ nhớ vật lý có thể không cần thiết. Một hữu ích cách tiếp cận sẽ là kết xuất nội dung của bộ nhớ tiến trình, sau đó sử dụng các công cụ khác để trích xuất thông tin cụ thể về quy trình, chẳng hạn như các mô-đun được tải, lệnh dòng được sử dụng để khởi chạy quá trình, hoặc mở tay cầm. Khi tất cả các thông tin được thu thập, bước tiếp theo có thể là lưu nội dung của cuộc hội thoại IM. Sau tất cả thích hợp thông tin đã được thu thập, tìm kiếm nội dung của bộ nhớ quá trình cho tàn dư của một cuộc trò chuyện trước đó hoặc dữ liệu khác có thể cung cấp cho bạn những manh mối hữu ích.

