**CUỐN SÁCH ĐEN NHỎ CỦA VIRUS MÁY TÍNH**

Cuốn sách đen nhỏ về vi-rút máy tính đã có năm năm tốt đẹp được xuất bản. Trong năm năm đó, nó đã mở ra một cánh cửa để nghiêm túc đặt câu hỏi liệu có nên công khai thông tin kỹ thuật về vi-rút máy tính hay không.

Khi tôi viết nó, phần lớn là một thử nghiệm. Tôi không biết điều gì sẽ xảy ra. Liệu mọi người có lấy các loại vi-rút mà nó chứa và viết lại chúng để tạo ra đủ loại vi-rút có sức tàn phá khủng khiếp không? Hay nói chung là chúng sẽ được sử dụng một cách có trách nhiệm? Vào thời điểm tôi viết, không có người chống vi-rút nào nói chuyện với tôi, và những gì tôi có thể tìm thấy trên báo in về chủ đề này phần lớn không gây ấn tượng theo quan điểm thực tế —rất nhiều sự cường điệu và gây hoang mang, nhưng rất ít nghiên cứu vững chắc có thể làm sáng tỏ những gì có thể xảy ra nếu tôi phát hành cuốn sách này. Là một người Mỹ yêu tự do và tìm kiếm kiến thức, tôi quyết định sẽ tiếp tục và làm điều đó—viết cuốn sách và xuất bản. Và tôi quyết định rằng nếu mọi người không sử dụng nó một cách có trách nhiệm, tôi sẽ thu hồi nó.

Năm năm sau, tôi phải nói rằng tôi tin chắc rằng cuốn sách này đã mang lại nhiều lợi ích hơn là tác hại.

Về mặt tích cực, rất nhiều người thực sự cần loại thông tin này—những người chịu trách nhiệm giữ cho máy tính không bị nhiễm vi-rút—giờ đã có thể có được thông tin đó. Trong khi những người dùng cá nhân có ít liên lạc với những người dùng máy tính khác có thể tự bảo vệ mình thành công bằng một phần mềm diệt vi-rút có sẵn, kinh nghiệm dường như chứng minh rằng điều đó không đúng khi một người bắt đầu xem xét mạng với 10.000 người dùng trên đó. Trước hết, rất ít hệ thống diệt vi-rút sẽ chạy trên 10.000 máy tính với nhiều cấu hình khác nhau, v.v. Thứ hai, khi ai đó trên mạng gặp phải vi-rút, họ phải có thể nói chuyện với ai đó trong tổ chức có kiến thức kỹ thuật chi tiết cần thiết để loại bỏ vi-rút theo cách hợp lý . Bạn không thể chỉ tắt một mạng lớn như vậy trong 4 ngày trong khi một ai đó từ đội ngũ hỗ trợ kỹ thuật của nhà cung cấp a-v của bạn được đưa đến để dọn dẹp hoặc phát hiện và phân tích vi-rút mới.

Thứ hai, những người chỉ quan tâm đến cách mọi thứ hoạt động cuối cùng cũng có thể tìm hiểu một chút về vi-rút máy tính. Thật khó để phủ nhận rằng chúng rất thú vị. Ý tưởng về một chương trình máy tính có thể cất cánh và có được cuộc sống hoàn toàn không phụ thuộc vào người tạo ra nó, thì, thật thú vị. Tôi nghĩ điều đó rất quan trọng. Rốt cuộc, nhiều phát minh thực sự hữu ích nhất không phải được tạo ra bởi các phòng thí nghiệm khổng lồ, bí mật, do chính phủ tài trợ, mà là bởi những cá nhân ngày này qua ngày khác tận tay làm ra một thứ gì đó. Họ nghĩ ra một cách để làm một điều gì đó tốt hơn, và thực hiện nó, và nó thay đổi thế giới. Tuy nhiên, điều đó sẽ không bao giờ xảy ra nếu bạn không thể có được thông tin cơ bản về cách một thứ gì đó hoạt động. Giống như việc tước đi chiếc búa của người thợ mộc rồi yêu cầu anh ta tìm ra cách xây dựng một tòa nhà tốt hơn.

Đồng thời, tôi phải thừa nhận rằng thí nghiệm mang tên The Little Black Book này không phải là không có nguy hiểm. Virus Stealth được mô tả trong các trang của nó đã thành công trong việc tự thiết lập trong tự nhiên, và tính đến ngày viết bài này, nó là #8 trong danh sách tần suất hàng năm, là sự kết hợp của các loại virus thường gặp nhất trong tự nhiên. Tôi rất tiếc vì nó đã tìm được đường vào trong tự nhiên, nhưng tôi vẫn tìm thấy ở đây một nét hài hước thiêng liêng hướng đến một số người chống vi-rút. Có khá nhiều lịch sử đằng sau loại virus này. Tôi sẽ chỉ đề cập đến nó một cách ngắn gọn vì tôi không muốn làm bạn chán với những cuộc chiến cá nhân của tôi. Trong lần in đầu tiên của The Little Black Book, Stealth được thiết kế để định dạng một rãnh bổ sung trên đĩa và ẩn mình ở đó. Tất nhiên, điều này chỉ hoạt động trên các máy có BIOS không kiểm tra số rãnh và những thứ tương tự như vậy— đặc biệt là trên các PC cũ. Và sau đó, nó không lây nhiễm đĩa mỗi khi chúng được truy cập. Điều này hạn chế khả năng sao chép của nó. Một số nhà phát triển phần mềm diệt vi-rút đã bình luận với tôi rằng họ nghĩ đây là một loại vi-rút tệ hại vì lý do đó, và gợi ý rằng tôi nên làm theo cách khác. Tôi đã do dự không muốn làm như vậy, tôi nói, vì tôi không muốn nó lây lan quá nhanh.

Không dừng lại ở việc đưa ra những gợi ý như vậy, một số người a-v này đã chỉ trích tôi trên báo in vì đã xuất bản những loại virus "tệ hại". Được thôi, tôi quyết định, nếu họ định chỉ trích cuốn sách như vậy, chúng tôi sẽ cải thiện các loại virus. Vòng tiếp theo tại nhà in, tôi đã cập nhật virus Stealth để hoạt động giống với Pakistani Brain hơn, ẩn các sector của nó trong các khu vực được đánh dấu là xấu trong bảng FAT và lây nhiễm nhanh như Stoned. Tuy nhiên, nó vẫn không ngăn được những lời chỉ trích ngớ ngẩn này Vào cuối năm ngoái, Robert Slade đã đánh giá cuốn sách này trong cuốn sách về virus của riêng mình và thấy nó còn thiếu sót vì các loại virus mà nó thảo luận không thực sự thành công trong việc lây lan. Ông ấy nghĩ rằng đây là lời chỉ trích khách quan. Từ ngày đó trở đi, có vẻ như Stealth không làm gì ngoài việc leo lên các bảng xếp hạng danh sách hoang dã. Việc kết hợp các kỹ thuật lây nhiễm hung hăng với một cơ chế tàng hình tốt thực sự đã chứng minh được hiệu quả... quá hiệu quả so với sở thích của tôi, để nói sự thật. Tôi chưa bao giờ có ý định viết ra những loại virus sẽ lọt vào danh sách các bảng xếp hạng hoang dã. Nhìn lại, tôi phải nói rằng tôi đã học được cách bỏ qua những lời chỉ trích ngu ngốc, ngay cả khi những kẻ ngốc muốn khiến tôi trông giống một kẻ ngốc so với sự khôn ngoan khó hiểu của họ.

Trong mọi trường hợp, Little Black Book đã có năm năm tốt như một ấn phẩm in. Tuy nhiên, với việc phát hành The Giant Black Book of Computer Viruses, nhà xuất bản đã quyết định ngừng xuất bản The Little Black Book. Họ đã đồng ý cung cấp phiên bản điện tử miễn phí, và đó là những gì bạn đang xem ngay bây giờ. Tôi hy vọng bạn sẽ thấy nó thú vị và bổ ích. Và nếu bạn thấy vậy, hãy xem danh mục đính kèm tại đây để biết thêm thông tin tuyệt vời về vi-rút từ nhà xuất bản.

**Giới thiệu**

Đây là cuốn đầu tiên trong bộ ba cuốn sách về virus máy tính. Trong những tập sách này, tôi muốn thách thức bạn suy nghĩ theo những cách mới về virus, và phá vỡ những khái niệm sai lầm và cách suy nghĩ sai lầm, và từ đó thảo luận về sự liên quan của virus máy tính trong thế giới ngày nay. Những cuốn sách này không phải là lời kêu gọi săn lùng phù thủy, hoặc hướng dẫn bảo vệ bản thân khỏi virus. Ngược lại, chúng sẽ hướng dẫn bạn cách thiết kế virus, triển khai chúng, và cải thiện chúng. Cả ba tập sách đều chứa đầy mã nguồn cho virus, bao gồm cả các loại mới và đã biết.

Không thể tránh khỏi việc những cuốn sách này sẽ làm mất lòng một số người. Thực tế, tôi hy vọng là như vậy. Họ cần phải làm vậy. Tôi tin rằng virus máy tính không phải là xấu xa và các lập trình viên có quyền tạo ra chúng, sở hữu chúng và thử nghiệm chúng. Kiểu lập trường đó sẽ làm mất lòng rất nhiều người, bất kể nó được trình bày như thế nào. Ngay cả một cách xử lý thuần túy về virus chỉ đơn giản là thảo luận về cách viết chúng và đưa ra một số ví dụ cũng sẽ là xúc phạm. Chỉ cần nghĩ đến một triệu tin tặc được trang bị vũ khí tốt cũng đủ khiến một số viên chức phát điên. Tuy nhiên, những cuốn sách này vượt ra ngoài cách xử lý mang tính kỹ thuật, để bảo vệ ý tưởng rằng virus có thể hữu ích, thú vị và chỉ đơn giản là vui vẻ. Điều đó chắc chắn sẽ chứng minh là xúc phạm hơn nữa. Tuy nhiên, sự thật vẫn là sự thật và cần phải được nói ra, ngay cả khi nó xúc phạm. Đạo đức và luân lý không thể được xác định bằng đa số phiếu bầu, cũng như chúng không thể được xác định bằng nòng súng hay một cái miệng to. Sức mạnh không tạo nên sự đúng đắn.

Nếu bạn trở thành một trong những người dễ bị xúc phạm hoặc khó chịu, hoặc nếu bạn thấy mình phản đối dữ dội với điều gì đó tôi nói, hãy nhớ lại những gì một người bạn có đầu óc thể thao của tôi đã từng nói với tôi: "Không đau, không có lợi". Câu đó ám chỉ đến việc xây dựng cơ bắp, nhưng nguyên tắc này cũng áp dụng cả về mặt trí tuệ cũng như thể chất. Nếu một người chỉ lắng nghe những người mà họ đồng ý, họ sẽ không bao giờ phát triển và họ sẽ không bao giờ thành công ngoài vòng tròn nhỏ bé của họ những người luôn đồng tình. Mặt khác, một người lắng nghe những ý tưởng khác nhau với nguy cơ bị xúc phạm, và ít nhất là cho rằng mình có thể sai, thì không thể không hưởng lợi từ điều đó. Vì vậy, nếu bạn bị xúc phạm bởi điều gì đó trong cuốn sách này, hãy chỉ trích—cả cuốn sách và bản thân bạn— và đừng rơi vào lối mòn và để người khác chỉ cho bạn cách suy nghĩ.

Ngay từ đầu, tôi muốn nhấn mạnh rằng tôi không ủng hộ bất kỳ ai ra ngoài và lây nhiễm hệ thống máy tính của bên vô tội bằng một loại vi-rút độc hại được thiết kế để phá hủy dữ liệu có giá trị hoặc làm hệ thống của họ dừng lại. Điều đó không chỉ sai mà còn là bất hợp pháp. Nếu bạn làm vậy, bạn có thể phải vào tù hoặc bị kiện hàng triệu đô la. Tuy nhiên, điều này không có nghĩa là việc tạo ra một loại vi-rút máy tính và thử nghiệm nó là bất hợp pháp, mặc dù tôi biết một số người muốn điều đó là bất hợp pháp. Tuy nhiên, nếu bạn tạo ra một loại vi-rút, hãy cẩn thận với nó. Hãy đảm bảo rằng bạn biết rằng nó đang hoạt động bình thường hoặc bạn có thể vô tình xóa sạch hệ thống của chính mình. Và hãy đảm bảo rằng bạn không vô tình phát tán nó ra thế giới, nếu không bạn có thể gặp phải vấn đề pháp lý ... ngay cả khi đó chỉ là một tai nạn. Người mất đi một năm làm việc có thể không tin rằng đó là một tai nạn. Và sớm thôi, việc lây nhiễm một hệ thống máy tính (kể cả hệ thống của chính bạn) bằng một loại vi-rút lành tính không gây hại gì cả có thể là bất hợp pháp. Từ khóa ở đây là trách nhiệm. Hãy có trách nhiệm. Nếu bạn làm điều gì đó mang tính phá hoại, hãy chuẩn bị chịu trách nhiệm. Các chương trình có trong cuốn sách này có thể nguy hiểm nếu sử dụng không đúng cách. Hãy đối xử với chúng bằng sự tôn trọng mà bạn dành cho một vũ khí gây chết người.

Tập đầu tiên trong ba tập này là phần giới thiệu kỹ thuật về những điều cơ bản khi viết virus máy tính. Tập này thảo luận về virus là gì và cách nó thực hiện chức năng của mình, đi sâu vào các thành phần chức năng chính của virus, từng bước một. Một số loại virus khác nhau được phát triển từ đầu, cung cấp cho người đọc thông tin thực tế về cách viết virus. Đó cũng là điều kiện tiên quyết để giải mã và hiểu bất kỳ loại virus nào mà người ta có thể gặp phải trong quá trình tính toán hàng ngày. Nhiều người nghĩ rằng virus là một loại nghệ thuật đen tối. Mục đích của tập này là đưa chúng ra khỏi bóng tối và xem xét chúng một cách thực tế, để xem chúng như chúng là gì, nói theo cách kỹ thuật: chương trình máy tính.

Tập thứ hai thảo luận về các ứng dụng khoa học của virus máy tính. Có một lĩnh vực nghiên cứu khoa học hoàn toàn mới được gọi là nghiên cứu sự sống nhân tạo (AL) đang mở ra do phát minh ra virus và các thực thể liên quan. Vì virus máy tính có chức năng tương tự như các sinh vật sống, nên sinh học có thể dạy chúng ta rất nhiều về chúng, cả cách chúng hoạt động và cách cải thiện chúng. Tuy nhiên, virus máy tính cũng có khả năng dạy chúng ta điều gì đó về các sinh vật sống. Chúng ta có thể tạo ra và kiểm soát virus máy tính theo cách mà chúng ta chưa thể kiểm soát được các sinh vật sống. Điều này cho phép chúng ta nhìn nhận cuộc sống một cách trừu tượng để tìm hiểu về bản chất thực sự của nó. Chúng ta thậm chí có thể suy ngẫm về những câu hỏi lớn như sự khởi đầu và quá trình tiến hóa tiếp theo của sự sống.

Tập thứ ba của loạt bài này thảo luận về các ứng dụng quân sự cho vi-rút máy tính. Người ta đều biết rằng vi-rút máy tính có thể cực kỳ phá hoại và chúng có thể được triển khai với rủi ro tối thiểu. Các tổ chức quân sự trên toàn thế giới cũng biết điều đó và coi khả năng tấn công bằng vi-rút là một mối đe dọa thực sự và một lựa chọn tấn công thực sự. Một số quan chức cấp cao ở nhiều quốc gia đã tin rằng máy tính của họ đã bị tấn công vì lý do chính trị. Vì vậy, tập thứ ba sẽ thăm dò các chiến lược quân sự và các cuộc tấn công trong đời thực, đồng thời đào sâu vào quá trình phát triển các hệ thống vũ khí vi-rút, đánh bại các biện pháp phòng thủ chống vi-rút, v.v.

Có thể bạn đang tự hỏi tại sao mình nên dành thời gian nghiên cứu những tập sách này. Xét cho cùng, virus máy tính rõ ràng không có giá trị thương mại nào ngoài các ứng dụng quân sự của chúng. Học cách viết chúng có thể không giúp bạn có khả năng được tuyển dụng hơn, hoặc cung cấp cho bạn các kỹ thuật mới để đưa vào các chương trình. Vậy tại sao lại lãng phí thời gian với chúng, trừ khi bạn cần chúng để gieo rắc sự hỗn loạn cho kẻ thù của mình? Hãy để tôi thử trả lời câu hỏi đó: Kể từ khi máy tính được phát minh vào những năm 1940, đã có một nhóm người tận tụy khám phá những khả năng vô hạn của những cỗ máy tuyệt vời này. Nhóm này bao gồm các nhà toán học và nhà khoa học nổi tiếng, cũng như hàng nghìn người đam mê vô danh đã tự chế tạo máy tính của riêng họ, và các lập trình viên. Có thể bạn đang tự hỏi tại sao mình nên dành thời gian nghiên cứu những tập sách này. Xét cho cùng, virus máy tính rõ ràng không có giá trị thương mại nào ngoài các ứng dụng quân sự của chúng. Học cách viết chúng có thể không giúp bạn có khả năng được tuyển dụng hơn, hoặc cung cấp cho bạn các kỹ thuật mới để đưa vào các chương trình. Vậy tại sao lại lãng phí thời gian với họ, trừ khi bạn cần họ gieo rắc sự hỗn loạn cho kẻ thù của bạn? Hãy để tôi thử trả lời câu hỏi đó: Kể từ khi máy tính được phát minh vào những năm 1940, đã có một nhóm người tận tụy khám phá những khả năng vô hạn của những cỗ máy tuyệt vời này. Nhóm này bao gồm các nhà toán học và nhà khoa học nổi tiếng, cũng như hàng ngàn người đam mê vô danh đã tự chế tạo máy tính của riêng họ và các lập trình viên.

Virus máy tính là một cách tiếp cận mới mang tính đột phá đối với ý tưởng này về “cỗ máy sống”. Thay vì cố gắng thiết kế thứ gì đó bắt chước hành vi phức tạp của con người, người ta bắt đầu bằng cách cố gắng sao chép những sinh vật sống đơn giản nhất. Những sinh vật đơn bào đơn giản không làm được gì nhiều. Những sinh vật nguyên thủy nhất lấy chất dinh dưỡng từ biển dưới dạng hóa chất vô cơ và lấy năng lượng từ mặt trời, và mục tiêu duy nhất của chúng rõ ràng là để tồn tại và sinh sản. Chúng không thông minh lắm, và sẽ rất khó để tranh luận về các khía cạnh siêu hình của chúng như “linh hồn”. Tuy nhiên, chúng làm những gì chúng được lập trình để làm, và chúng làm rất hiệu quả. Nếu chúng ta cố gắng bắt chước những sinh vật như vậy bằng cách chế tạo một cỗ máy— một con rô-bốt nhỏ—đi khắp nơi thu thập nguyên liệu thô và lắp ráp chúng lại để tạo ra một con rô-bốt nhỏ khác, chúng ta sẽ có một nhiệm vụ rất khó khăn. Mặt khác, hãy nghĩ đến một vũ trụ hoàn toàn mới—không phải thế giới vật lý này, mà là thế giới điện tử, tồn tại bên trong một chiếc máy tính. Đây là thế giới của con vi-rút. Ở đây, nó có thể “sống” theo nghĩa không quá khác biệt so với sự sống sinh học nguyên thủy. Virus máy tính có cùng mục tiêu như một sinh vật sống—là tồn tại và sinh sản. Nó có những trở ngại về môi trường cần vượt qua, có thể “giết chết” nó và khiến nó không hoạt động. Và một khi được giải phóng, nó dường như có một tâm trí riêng. Nó chạy trong thế giới điện tử của nó và làm những gì nó được lập trình để làm. Theo nghĩa này, nó rất sống động.

Không còn nghi ngờ gì nữa, sự khởi đầu của sự sống là một cột mốc quan trọng trong lịch sử của trái đất. Tuy nhiên, nếu chúng ta cố gắng xem xét nó từ quan điểm của vật chất vô tri, thì thật khó để tưởng tượng sự sống là nhiều hơn một sự phiền toái. Chúng ta thường cho rằng sự sống là tốt và nó xứng đáng được bảo vệ. Tuy nhiên, chúng ta không thể lùi lại một bước nữa và coi sự sống là có lợi cho thế giới vô tri. Nếu chúng ta chỉ xem xét các nguyên tử của vũ trụ, thì có gì khác biệt nếu nhiệt độ là bảy mươi độ farenheit hay hai mươi triệu? Có gì khác biệt nếu trái đất được bao phủ bởi các vật liệu phóng xạ? Không có gì cả. Bất cứ khi nào chúng ta nói về môi trường và sinh thái, chúng ta luôn cho rằng sự sống là tốt và nó nên được nuôi dưỡng và bảo tồn. Các sinh vật sống trên khắp thế giới vô tri vô giác mà không mấy quan tâm đến nó, từ tế bào nhỏ nhất tự do thu thập các chất dinh dưỡng cần thiết và làm ô nhiễm nguồn nước mà nó bơi vào, cho đến người đàn ông nghiền nát đá để tinh chế kim loại từ chúng và chế tạo máy bay. Các sinh vật sống sử dụng thế giới vật chất theo cách chúng thấy phù hợp. Ngay cả khi mọi người tức giận về điều gì đó như khai thác lộ thiên, hoặc sự cố tràn dầu, thì điểm tham chiếu của họ không phải là bản chất vô tri vô giác. Đó là một khái niệm hoàn toàn ích kỷ (đối với sự sống) thúc đẩy họ. Khai thác làm hỏng vẻ đẹp của cảnh quan—một vẻ đẹp nằm trong mắt của người xem (sống)— và khiến nơi đó trở nên không thể ở được. Nếu một người không đặc biệt chú trọng đến sự sống, thì người đó cũng có thể thúc đẩy khai thác lộ thiên như một nỗ lực đưa trái đất trở lại trạng thái tiền sinh học!

Tôi nói tất cả những điều này không phải vì tôi có vấn đề gì với các nhà sinh thái học. Thay vào đó, tôi muốn áp dụng lý luận tương tự vào thế giới của virus máy tính. Miễn là người ta chỉ sử dụng các tiêu chí tài chính để đánh giá giá trị của một chương trình máy tính, thì virus chỉ có thể được coi là mối đe dọa. Chúng làm gì ngoài việc phá hủy các chương trình và dữ liệu có giá trị? Chúng tàn nhẫn trong việc cố gắng truy cập vào các tài nguyên hệ thống máy tính, và thường thì chúng càng tàn nhẫn, thì càng thành công. Nhưng điều đó khác gì với sự sống sinh học? Nếu một cụm rêu có thể tấn công một tảng đá để có chút ánh nắng mặt trời và phát triển, thì nó sẽ làm như vậy một cách tàn nhẫn. Chúng ta gọi đó là đẹp. Vậy thì điều đó khác gì với một virus máy tính tự gắn vào một chương trình? Nếu tất cả những gì người ta quan tâm là bảo tồn các vật thể vô tri (là các chương trình thông thường) trong thế giới điện tử này, thì tất nhiên virus là một mối phiền toái.

Tuy nhiên, điều quan trọng nhất đối với bạn là gì. Có vẻ như nền văn hóa hiện đại đã thoái hóa đến mức hầu hết đàn ông không có mục tiêu nào cao hơn trong cuộc sống ngoài việc tìm kiếm hòa bình và thịnh vượng cho riêng mình. Khi nói đến hòa bình cá nhân, tôi không có ý nói đến tự do khỏi chiến tranh, mà là tự do suy nghĩ và tin vào bất cứ điều gì bạn muốn mà không bao giờ bị thách thức trong đó. Nói thẳng thắn hơn, đó là tự do sống trong thế giới tưởng tượng do chính bạn tạo ra. Khi nói đến thịnh vượng, tôi chỉ muốn nói đến sự giàu có về vật chất ngày càng tăng. Karl Marx đã nhìn vào toàn thể nhân loại và nói rằng động lực thúc đẩy đằng sau mỗi người đàn ông là sự thịnh vượng về kinh tế của họ. Ông nói rằng kết quả là toàn bộ lịch sử có thể được diễn giải theo các thuật ngữ đấu tranh giai cấp—mọi người đấu tranh giành quyền kiểm soát kinh tế. Mặc dù nhiều người trong chính phủ của chúng ta chỉ trích Marx là cha đẻ của chủ nghĩa cộng sản, nhưng quốc gia của chúng ta đang cố gắng chen vào chiếc áo bó mà ông đã dành cho chúng ta. Đó là lý do tại sao hai trong số những lời hứa quan trọng nhất trong chiến dịch tranh cử của George Bush là "thêm bốn năm thịnh vượng nữa" và "không có thuế mới". Mọi người bỏ phiếu bằng ví tiền của mình, ngay cả khi họ biết các chính trị gia đang nói dối trắng trợn.

Trong một xã hội có những giá trị như vậy, máy tính trở thành một nguồn tài nguyên mà mọi người sử dụng để khai thác lượng thông tin dồi dào và thao túng chúng theo hướng có lợi cho họ. Nếu đó là tất cả những gì máy tính có, thì vi-rút máy tính là một mối phiền toái, và chúng cần phải bị loại bỏ. Chắc chắn phải có một mục đích cao cả hơn cho nhân loại hơn là kiếm tiền, mặc dù điều đó có thể cần thiết. Marx có thể không nghĩ như vậy. Chính phủ có thể không nghĩ như vậy. Và rất nhiều người lắm mồm có thể không nghĩ như vậy. Tuy nhiên, những người vĩ đại từ mọi thời đại và mọi quốc gia đều chứng minh cho sự thật rằng con người có một mục đích cao cả hơn. Chúng ta không nên giống như Socrates, người tự cho mình là ngu dốt, và người đã tìm kiếm Chân lý và Trí tuệ, và coi trọng chúng hơn cả bạc và vàng sao? Và nếu vậy, câu hỏi thực sự quan trọng không phải là máy tính có thể giúp chúng ta giàu có hay trao cho chúng ta quyền lực đối với người khác như thế nào, mà là chúng có thể giúp chúng ta trí tuệ như thế nào. Chúng ta có thể học được gì về bản thân mình? về thế giới của chúng ta? và, vâng, thậm chí có thể về Chúa? Một khi chúng ta tập trung vào điều đó, vi-rút máy tính trở nên rất thú vị. Liệu chúng ta có thể hiểu cuộc sống tốt hơn một chút nếu chúng ta có thể tạo ra thứ gì đó tương tự, nghiên cứu nó và cố gắng hiểu nó không? Và nếu chúng ta hiểu cuộc sống tốt hơn, liệu chúng ta có hiểu cuộc sống của mình và thế giới của mình tốt hơn không?

Trước tiên, xin lưu ý: Nhiều thế kỷ trước, quốc gia của chúng ta được thành lập dựa trên các nguyên tắc triết học về chính quyền tốt, được nêu rõ trong Tuyên ngôn Độc lập và Hiến pháp. Khi hòa bình và thịnh vượng của cá nhân trở nên quan trọng hơn các nguyên tắc về chính quyền tốt, các nguyên tắc này đã bị thao túng và định nghĩa lại để phù hợp với ý thích của những người cầm quyền. Chính phủ ngày càng ít nhạy cảm hơn với các quyền dân sự, trong khi các nhóm lợi ích chính trị và tài chính khác nhau dễ dàng thao túng các nhà lãnh đạo của chúng ta để có lợi cho họ.

Vì mọi người phần lớn đã ngừng thách thức lẫn nhau về những gì họ tin tưởng, thay vào đó chấp nhận ý tưởng rằng bất cứ điều gì bạn muốn tin đều ổn, nên chính phủ không còn có thể khiến mọi người tuân thủ luật pháp nữa vì mọi người đều tin vào một tập hợp các nguyên tắc nhất định mà luật pháp dựa trên. Do đó, chính phủ phải ép buộc mọi người tuân thủ luật pháp bằng các hình phạt ngày càng khắc nghiệt hơn đối với hành vi bất tuân—những hình phạt thường đi ngược lại các quyền dân sự đã được thiết lập từ lâu. Hơn nữa, chính phủ phải hạn chế khả năng tìm kiếm sự cứu trợ của người dân trung bình. Ví dụ, rất phổ biến khi chính phủ chà đạp lên mọi quyền hiến định lâu đời khi thực thi luật thuế. IRS thường xuyên buộc hàng trăm nghìn người phải làm chứng chống lại chính họ. IRS thường xuyên đặt gánh nặng chứng minh lên bị cáo, tịch thu tài sản của bị cáo mà không cần xét xử, v.v., v.v. Điểm mấu chốt là chính phủ không nên thu tiền từ công dân của mình nếu phải chứng minh các giấy tờ thuế của họ là sai. Toàn bộ hệ thống sẽ sụp đổ trong tình trạng quá tải nghiêm trọng. Về mặt kinh tế, tốt hơn hết là đưa gánh nặng chứng minh lên vai công dân, bất kể có Tuyên ngôn Nhân quyền hay không.

Tương tự như vậy, việc thách thức chính phủ về một vấn đề quyền là điều gần như không thể, trừ khi trường hợp của bạn tình cờ phục vụ mục đích của một nhóm lợi ích đặc biệt quyền lực nào đó. Trong một phòng xử án tiêu chuẩn, người ta thường không thể đưa ra chủ đề về quyền hiến định. Câu hỏi duy nhất cần tranh luận là liệu một số luật cụ thể có bị vi phạm hay không. Việc kháng cáo lên Tòa án Tối cao sẽ tốn hàng triệu đô la, nếu các thẩm phán có động cơ chính trị thậm chí còn hạ mình để thụ lý vụ án. Vì vậy, chính phủ trở nên gần như toàn năng, Chúa bước đi trên trái đất, đối với người dân thường. Một người dường như không có nhiều cách giải quyết ngoài việc tuân theo một cách mù quáng những người có quyền lực.

Khi chúng ta bắt đầu nói về vi-rút máy tính, chúng ta đang đi vào một số vấn đề mà một số người muốn đặt biển báo "Cấm xâm phạm". Quốc hội Hoa Kỳ đã xem xét một "Đạo luật diệt vi-rút máy tính" sẽ biến việc viết vi-rút thành trọng tội hoặc hai bên tự nguyện trao đổi vi-rút. Không cần quan tâm đến việc Hiến pháp bảo đảm quyền tự do ngôn luận và tự do báo chí. Không cần quan tâm đến việc nó bảo đảm cho công dân quyền mang vũ khí quân sự (và vi-rút có thể được phân loại như vậy). Mặc dù luật đó chưa được thông qua cho đến thời điểm viết bài này, nhưng có thể sẽ được thông qua vào thời điểm bạn đọc cuốn sách này. Nếu vậy, tôi sẽ nói mà không do dự rằng đó là một chế độ chuyên chế khốn khổ, nhưng là chế độ mà chúng ta không thể làm gì nhiều... vào lúc này.

Một số nhà lãnh đạo của chúng ta có thể lập luận rằng nhiều người không có khả năng xử lý trách nhiệm về quyền lực đi kèm với việc hiểu biết về vi-rút máy tính, cũng giống như họ lập luận rằng mọi người không thể xử lý quyền lực sở hữu súng trường tấn công hoặc súng máy. Có lẽ một số người không thể. Nhưng tôi tự hỏi, liệu các nhà lãnh đạo của chúng ta có khả năng xử lý tốt hơn nhiều các loại vũ khí nguy hiểm hơn của luật pháp và sức mạnh vô hạn không? Rõ ràng là họ nghĩ vậy, vì họ đang bận rộn cố gắng tập trung mọi quyền lực vào tay mình. Tôi không đồng ý. Nếu những người trong chính phủ có thể xử lý được quyền lực, thì cá nhân cũng vậy. Nếu cá nhân không thể, thì những người đại diện của họ cũng không thể, và mục đích của chúng ta là hoặc là chuyên chế hoặc hỗn loạn. Vì vậy, không có gì sai khi cố gắng khôi phục một số quyền lực nhỏ cho cá nhân.

Nhưng hãy nhớ rằng: những người tìm kiếm sự thật và những người thông thái đã bị những kẻ ngốc quyền lực ngược đãi ở mọi thời đại. Mặc dù virus máy tính có thể rất thú vị và đáng giá, nhưng những người quan tâm đến chúng có thể phải đối mặt với một số thách thức nghiêm trọng từ những kẻ hèn hạ. Vì vậy, hãy cẩn thận.

Bây giờ hãy cùng tôi và có thái độ của những nhà khoa học đầu tiên. Những nhà thám hiểm này muốn hiểu thế giới vận hành như thế nào—và việc nó có thể biến thành lợi nhuận hay không không quan trọng. Họ cố gắng trở nên khôn ngoan hơn trong những điều thực sự quan trọng bằng cách hiểu thế giới tốt hơn một chút. Rốt cuộc, có giá trị gì khi chế tạo một chiếc kính thiên văn để bạn có thể nhìn thấy các mặt trăng xung quanh Sao Mộc? Galileo hẳn đã nhìn thấy điều gì đó trong đó, và điều đó hẳn có ý nghĩa đủ để ông đứng lên chống lại những nhà cầm quyền thời bấy giờ và làm điều đó, và nói về điều đó, và khuyến khích những người khác làm điều đó. Và phải vào tù vì điều đó. Ngày nay, một số người vui mừng vì ông đã làm như vậy.

Vậy tại sao không có cùng thái độ khi nói đến việc tạo ra sự sống trên máy tính? Người ta phải tự hỏi điều đó có thể dẫn đến đâu. Có thể có một thế giới hoàn toàn mới về các dạng sống điện tử khả thi, trong đó virus máy tính chỉ là loại thô sơ nhất? Có lẽ chúng là dạng tương tự điện tử của tế bào đơn giản nhất Vậy tại sao không có cùng thái độ khi nói đến việc tạo ra sự sống trên máy tính? Người ta phải tự hỏi điều đó có thể dẫn đến đâu. Có thể có một thế giới hoàn toàn mới về các dạng sống điện tử khả thi, trong đó virus máy tính chỉ là loại thô sơ nhất? Có lẽ chúng là dạng tương tự điện tử của tế bào đơn giản nhất.

Có điều gì đó ở một số người đàn ông chỉ đơn giản thúc đẩy họ khám phá những điều chưa biết. Khi đứng ở rìa của một đại dương bao la mà chưa từng có con tàu nào đi qua, thật khó để không tự hỏi điều gì nằm ngoài đường chân trời chỉ vì những người cai trị thời đó nói với bạn rằng bạn sẽ rơi khỏi rìa thế giới (hoặc họ sẽ đẩy bạn xuống) nếu bạn cố gắng tìm hiểu. Có lẽ họ đúng. Có lẽ không có gì có giá trị ngoài kia. Tuy nhiên, những nhà thám hiểm vĩ đại khác qua nhiều thời đại đã khám phá các đại dương khác và thành công. Và một điều chắc chắn là: chúng ta sẽ không bao giờ biết nếu ai đó không nhìn. Vì vậy tôi muốn mời bạn leo lên chiếc bè nhỏ này mà tôi đã đóng và đi khám phá. . . .

**Những điều cơ bản về Virus máy tính**

Một loạt các bài báo và sách tiêu cực đã gây ra một loại bệnh hoang tưởng mới ở những người dùng máy tính: nỗi sợ hãi vô lý về vi-rút máy tính. Bệnh hoang tưởng này có thể xảy ra vì :

a) máy tính là những cỗ máy rất phức tạp, thường hoạt động theo những cách mà người dùng trung bình không nhận ra và

b) vi-rút máy tính vẫn cực kỳ hiếm. Do đó, hầu hết người dùng máy tính chưa bao giờ gặp phải cuộc tấn công của vi-rút máy tính. Trải nghiệm duy nhất của họ là những gì họ đã đọc hoặc nghe (và chỉ những vấn đề tồi tệ nhất mới được đưa vào bản in). Sự kết hợp giữa sự thiếu hiểu biết, thiếu kinh nghiệm và các báo cáo gây sợ hãi về mối nguy hiểm này chính là công thức hoàn hảo cho chứng cuồng loạn tập thể.

Hầu hết các vấn đề mà mọi người gặp phải với máy tính chỉ đơn giản là lỗi của chính họ. Ví dụ, họ vô tình xóa tất cả các tệp trong thư mục hiện tại của họ thay vì trong một thư mục khác, như họ dự định, hoặc họ định dạng sai đĩa. Hoặc có lẽ ai đó thường xuyên làm điều gì đó sai trái vì thiếu hiểu biết, chẳng hạn như tắt máy tính giữa chừng một chương trình, khiến các tệp bị xáo trộn. Tiếp theo sau những loại vấn đề này là các vấn đề về phần cứng, chẳng hạn như ổ đĩa mềm không thẳng hàng hoặc lỗi ổ cứng. Những vấn đề thường gặp như vậy trở nên tồi tệ hơn mức cần thiết khi người dùng không lập kế hoạch cho chúng và không sao lưu công việc của họ thường xuyên. Sự ngu ngốc này có thể dễ dàng biến một vấn đề có thể tốn 300 đô la cho một ổ cứng mới thành một cơn ác mộng mà cuối cùng sẽ tốn hàng chục nghìn đô la. Khi một thảm họa như vậy xảy ra, bản chất con người là muốn tìm ai đó hoặc điều gì đó khác để đổ lỗi, thay vì thừa nhận đó là lỗi của chính mình. Vi-rút đã được chứng minh là một vật tế thần tuyệt vời cho mọi loại vấn đề.

Tất nhiên, có những lúc mọi người muốn phá hủy máy tính. Trong thời chiến, một quốc gia có thể muốn làm tê liệt kẻ thù của mình bằng cách phá hủy cơ sở dữ liệu tình báo của họ. Nếu một nhân viên bị chủ nhân đối xử tệ, anh ta có thể muốn trả đũa, và anh ta có thể không thể có được sự hỗ trợ pháp lý. Người ta cũng có thể tưởng tượng một nhà nước toàn trị cố gắng kiểm soát mọi hành động của công dân bằng máy tính, và một nhóm người tốt cố gắng ngăn chặn điều đó. Mặc dù người ta có thể đập một máy tính, hoặc phá hủy dữ liệu của nó, nhưng không phải lúc nào người ta cũng có quyền truy cập vào máy sẽ là mục tiêu của cuộc tấn công. Vào những lúc khác, người ta có thể không thể thực hiện một cuộc tấn công vật lý mà không phải đối mặt với sự phát hiện và truy tố nhất định. Mặc dù một cuộc tấn công vô cớ, và thậm chí là trả thù, có thể không đúng, nhưng mọi người vẫn chọn những con đường như vậy (và ngay cả một cuộc tấn công hoàn toàn mang tính phòng thủ chắc chắn cũng bị coi là sai bởi một kẻ xâm lược kiêu ngạo). Tuy nhiên, đối với một lập trình viên lão luyện, việc tiếp cận vật lý với máy là không cần thiết để làm tê liệt nó.

Những người đã tấn công máy tính và dữ liệu của chúng đã phát minh ra một số loại chương trình khác nhau. Vì rõ ràng là phải che giấu bản chất phá hoại của một chương trình để lừa ai đó thực hiện chương trình đó, nên các thủ thuật lừa đảo là điều bắt buộc trong trò chơi này. Thủ thuật đầu tiên và lâu đời nhất là "con ngựa thành Troy". Con ngựa thành Troy có thể có vẻ là một chương trình hữu ích, nhưng thực tế là nó có tính phá hoại. Nó dụ bạn thực hiện chương trình đó vì nó hứa hẹn sẽ là một chương trình có giá trị cho máy tính của bạn—những cách mới và tốt hơn để làm cho máy của bạn hiệu quả hơn—nhưng khi bạn thực hiện chương trình, thật bất ngờ! Thứ hai, mã phá hoại có thể được ẩn dưới dạng "quả bom logic" bên trong một chương trình hữu ích khác. Bạn sử dụng chương trình thường xuyên và nó hoạt động tốt. Tuy nhiên, khi một sự kiện nào đó xảy ra, chẳng hạn như một ngày nào đó trên đồng hồ hệ thống, quả bom logic sẽ "phát nổ" và gây thiệt hại. Các chương trình này được thiết kế đặc biệt để phá hủy khai bởi tác giả của chúng hoặc một cộng sự tự nguyện trên hệ thống máy tính sẽ là đối tượng của cuộc tấn công.

Luôn có nguy cơ đối với thủ phạm của hành vi phá hoại như vậy. Bằng cách nào đó, anh ta phải triển khai mã phá hoại trên máy mục tiêu mà không bị phát hiện. Nếu điều đó có nghĩa là anh ta phải tự mình đưa chương trình vào máy hoặc đưa cho người dùng không nghi ngờ, anh ta sẽ gặp rủi ro. Rủi ro có thể khá nhỏ, đặc biệt là nếu thủ phạm thường có quyền truy cập vào các tệp trên hệ thống, nhưng rủi ro của anh ta không bao giờ bằng không.

Với những rủi ro đáng kể như vậy, có một động lực mạnh mẽ để phát triển các cơ chế triển khai xảo quyệt nhằm đưa mã phá hoại vào hệ thống máy tính. Việc triển khai không thể theo dõi là chìa khóa để tránh bị đưa ra xét xử vì tội phản quốc, gián điệp hoặc phá hoại. Trong số những lập trình viên máy tính tinh vi nhất, vi-rút máy tính là phương tiện được lựa chọn để triển khai mã phá hoại. Đó là lý do tại sao vi-rút gần như đồng nghĩa với sự phá hoại bừa bãi.

Tuy nhiên, chúng ta phải nhận ra rằng vi-rút máy tính không có bản chất phá hoại. Đặc điểm thiết yếu của một chương trình máy tính khiến nó được phân loại là vi-rút không phải là khả năng phá hủy dữ liệu, mà là khả năng kiểm soát máy tính và tạo ra một bản sao hoàn chỉnh của chính nó. Nó có thể tự sao chép. Khi được thực thi, nó tạo ra một hoặc nhiều bản sao của chính nó. Những bản sao đó sau đó có thể được thực thi, để tạo ra nhiều bản sao hơn nữa, vô tận. Không phải tất cả các chương trình máy tính có tính phá hoại đều được phân loại là vi-rút vì chúng không phải tất cả đều tự sao chép, và không phải tất cả vi-rút đều có tính phá hoại vì việc tự sao chép không phải là phá hoại. Tuy nhiên, tất cả vi-rút đều có thể tự sao chép. Tuy nhiên, ý tưởng cho rằng vi-rút máy tính luôn có tính phá hoại đã ăn sâu vào suy nghĩ của hầu hết mọi người. Bản thân thuật ngữ "vi-rút" là một cách gọi không chính xác và mang tính cảm xúc. Thuật ngữ khoa học chính xác cho vi-rút máy tính là "máy tự động tự sao chép", hoặc viết tắt là "SRA". Thuật ngữ này mô tả chính xác những gì một chương trình như vậy làm, thay vì gắn năng lượng cảm xúc vào đó. Tuy nhiên, chúng tôi sẽ tiếp tục sử dụng thuật ngữ “virus” trong toàn bộ cuốn sách này, trừ khi chúng tôi thảo luận về virus máy tính (SRA) và virus sinh học cùng lúc, và chúng tôi cần làm rõ sự khác biệt.

Nếu ai đó cố gắng đưa ra sự tương tự giữa thế giới điện tử của các chương trình và byte bên trong máy tính và thế giới vật lý mà chúng ta biết, thì vi-rút máy tính là một sự tương tự rất gần với đơn vị sinh học đơn giản nhất của sự sống, một sinh vật quang hợp đơn bào. Bỏ qua những câu hỏi siêu hình như "linh hồn", một sinh vật sống có thể được phân biệt với vật không sống ở chỗ nó dường như có hai mục tiêu: (a) tồn tại và (b) sinh sản. Mặc dù người ta có thể nêu ra những câu hỏi siêu hình chỉ bằng cách nói rằng một sinh vật sống có "mục tiêu", nhưng chúng chắc chắn có vẻ như vậy, nếu người ngoài cuộc chưa được giáo dục thoát khỏi lối suy nghĩ đó. Và chắc chắn ý tưởng về một mục tiêu sẽ áp dụng cho một chương trình máy tính, vì nó được viết bởi một người có mục đích trong đầu. Vì vậy, theo nghĩa này, một vi-rút máy tính có cùng hai mục tiêu như một sinh vật sống: tồn tại và sinh sản. Những sinh vật sống đơn giản nhất chỉ phụ thuộc vào môi trường vô tri, vô cơ để đạt được mục tiêu của chúng. Chúng lấy nguyên liệu thô từ môi trường xung quanh và sử dụng năng lượng từ mặt trời để tổng hợp bất kỳ hóa chất nào chúng cần để thực hiện công việc. Sinh vật không phụ thuộc vào một dạng sống khác mà nó phải ăn hoặc tấn công để tiếp tục tồn tại. Tương tự như vậy, vi-rút máy tính sử dụng các tài nguyên của hệ thống máy tính như bộ nhớ đĩa và thời gian CPU để đạt được mục tiêu của mình. Cụ thể, nó không tấn công các máy tự động tự sinh sản khác và “ăn” chúng theo cách tương tự như vi-rút sinh học. Thay vào đó, vi-rút máy tính là đơn vị sống đơn giản nhất trong thế giới điện tử này bên trong máy tính. (Tất nhiên, có thể hình dung rằng người ta có thể viết một chương trình phức tạp hơn có thể hoạt động giống như một vi-rút sinh học và tấn công các SRA khác.)

Trước khi máy tính cá nhân ra đời, phạm vi điện tử mà vi-rút máy tính có thể "sống" là cực kỳ hạn chế. Máy tính rất hiếm và chúng có nhiều loại CPU và hệ điều hành khác nhau. Vì vậy, một người thợ có thể đã viết một vi-rút và để nó thực thi trên hệ thống của mình. Tuy nhiên, sẽ hiếm có nguy cơ vi-rút thoát ra và lây nhiễm sang các máy khác. Vi-rút vẫn nằm trong tầm kiểm soát của chủ nhân. Tuy nhiên, thời đại máy tính sản xuất hàng loạt đã mở ra một thế giới hoàn toàn mới cho vi-rút. Hàng triệu máy móc trên khắp thế giới, tất cả đều có cùng kiến trúc và hệ điều hành cơ bản khiến vi-rút máy tính có thể thoát ra và bắt đầu cuộc sống riêng của mình. Vi-rút có thể nhảy từ máy này sang máy khác, hoàn thành các mục tiêu được lập trình sẵn, không ai kiểm soát và ít người có thể ngăn chặn vi-rút. Và vì vậy, vi-rút đã trở thành một dạng sống điện tử khả thi vào những năm 1980.automaton có thể được mô phỏng trên máy tính. Tuy nhiên, đó không phải là chương trình có thể chạy trực tiếp trên bất kỳ máy tính nào được biết đến vào thời của von Neumann. Tương tự như vậy, người ta có thể viết một chương trình chỉ đơn giản sao chép chính nó vào một tệp khác. Ví dụ, "1.COM" có thể tạo ra “2.COM” sẽ là bản sao chính xác của chính nó (cả hai tệp chương trình trên máy kiểu IBM PC.) Vấn đề với những hỗn hợp như vậy là khả năng tồn tại. Sự tồn tại liên tục của chúng hoàn toàn phụ thuộc vào người đàn ông ở bảng điều khiển. Một phiên bản tinh vi hơn của chương trình như vậy có thể dựa vào việc lừa dối người đàn ông đó ở bảng điều khiển để tự lan truyền. Chương trình này được gọi là sâu. Virus máy tính vượt qua rào cản kiểm soát của người vận hành bằng cách ẩn mình trong các chương trình khác. Do đó, nó có thể truy cập vào CPU chỉ vì mọi người chạy các chương trình mà nó tình cờ tự gắn vào mà họ không biết. Khả năng tự gắn vào các chương trình khác là điều khiến vi-rút trở thành một dạng sống điện tử khả thi. Đó là điều khiến nó tự xếp vào một lớp. Thực tế là vi-rút máy tính tự gắn vào các chương trình khác khiến nó được gọi là “vi-rút”. Tuy nhiên, phép loại suy đó là sai vì các chương trình mà nó gắn vào không hề có sự sống.

**Các loại vi-rút**

Virus máy tính có thể được phân loại thành nhiều loại khác nhau. Loại đầu tiên và phổ biến nhất là loại virus lây nhiễm bất kỳ chương trình ứng dụng nào. Trên PC IBM và các bản sao chạy dưới PC-DOS hoặc MS-DOS, hầu hết các chương trình và dữ liệu không thuộc hệ điều hành đều được lưu trữ dưới dạng tệp. Mỗi tệp có tên tệp dài tám ký tự và phạm vi dài ba ký tự. Một tệp thông thường có thể được gọi là “TRUE.TXT”, trong đó “TRUE” là tên và “TXT” là phạm vi. Phạm vi thường cung cấp một số thông tin về bản chất của tệp—trong trường hợp này “TRUE.TXT” có thể là tệp văn bản. Các chương trình phải luôn có phạm vi là “COM”, “EXE” hoặc “SYS”. Trong DOS, chỉ các tệp có phạm vi này mới có thể được bộ xử lý trung tâm thực thi. Nếu người dùng cố gắng thực thi bất kỳ loại tệp nào khác, DOS sẽ tạo ra lỗi và từ chối nỗ lực thực thi tệp.

Vì mục tiêu của vi-rút là được máy tính thực thi, nên vi-rút phải tự gắn vào tệp COM, EXE hoặc SYS. Nếu vi-rút gắn vào bất kỳ tệp nào khác, vi-rút có thể làm hỏng một số dữ liệu, nhưng thông thường vi-rút sẽ không được thực thi và không thể tự sao chép. Vì mỗi loại tệp thực thi này có cấu trúc khác nhau, nên vi-rút phải được thiết kế để tự gắn vào một loại tệp cụ thể. Vi-rút được thiết kế để tấn công các tệp COM không thể tấn công các tệp EXE và ngược lại, cũng như không thể tấn công các tệp SYS. Tất nhiên, người ta có thể thiết kế một vi-rút có thể tấn công hai hoặc thậm chí ba loại tệp, nhưng sẽ cần một phương pháp sao chép riêng cho từng loại tệp.

Loại virus chính tiếp theo tìm cách tự gắn vào một tệp cụ thể, thay vì tấn công bất kỳ tệp nào thuộc một loại nhất định. Do đó, chúng ta có thể gọi nó là virus dành riêng cho ứng dụng. Những loại virus này sử dụng kiến thức chi tiết về các tệp mà chúng tấn công để ẩn tốt hơn so với khả năng xâm nhập vào bất kỳ tệp nào. Ví dụ, chúng có thể ẩn trong một vùng dữ liệu bên trong chương trình thay vì kéo dài tệp. Tuy nhiên, để làm được điều đó, virus phải biết vùng dữ liệu nằm ở đâu trong chương trình và điều đó khác nhau tùy theo từng chương trình.

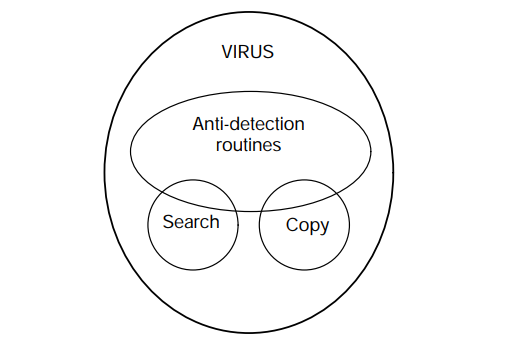
Loại virus thứ hai này thường tập trung vào các tệp liên quan đến DOS, như COMMAND.COM, vì chúng có trên hầu như mọi máy tính hiện có. Tuy nhiên, bất kể tệp virus nào tấn công, thì nó phải rất, rất phổ biến, nếu không virus sẽ không bao giờ có thể tìm thấy một bản sao khác của tệp đó để tái tạo, và do đó, nó sẽ không đi đến đâu cả. Chỉ với một tệp như COMMAND.COM, thì mới có thể bắt đầu nhảy từ máy này sang máy khác và du hành khắp thế giới.

Loại vi-rút cuối cùng được gọi là "vi-rút boot sector". Vi-rút này là một dạng cải tiến hơn nữa của vi-rút dành riêng cho ứng dụng, tấn công một vị trí cụ thể trên ổ đĩa của máy tính, được gọi là boot sector. Boot sector là thứ đầu tiên máy tính tải vào bộ nhớ từ đĩa và thực thi khi được bật. Bằng cách tấn công vào khu vực này của đĩa, vi-rút có thể kiểm soát máy tính ngay lập tức, mỗi khi máy được bật, trước khi bất kỳ chương trình nào khác có thể thực thi. Theo cách này, vi-rút có thể thực thi trước khi bất kỳ chương trình hoặc người nào khác có thể phát hiện ra sự tồn tại của nó.

**Các thành phần chức năng của một loại vi-rút**

Mỗi loại vi-rút máy tính khả thi phải có ít nhất hai phần cơ bản, hoặc chương trình con, nếu nó được gọi là vi-rút. Đầu tiên, nó phải chứa một chương trình tìm kiếm, chương trình này sẽ định vị các tệp hoặc vùng mới trên đĩa là mục tiêu đáng để lây nhiễm. Chương trình này sẽ xác định mức độ sinh sản của vi-rút, ví dụ, vi-rút sinh sản nhanh hay chậm, có thể lây nhiễm nhiều đĩa hay một đĩa duy nhất, và có thể lây nhiễm mọi phần của đĩa hay chỉ một số vùng cụ thể. Cũng như tất cả các chương trình, có sự đánh đổi giữa kích thước và chức năng ở đây. Chương trình tìm kiếm càng tinh vi, thì càng chiếm nhiều không gian. Vì vậy, mặc dù một chương trình tìm kiếm hiệu quả có thể giúp vi-rút lây lan nhanh hơn, nhưng nó sẽ khiến vi-rút lớn hơn, và điều đó không phải lúc nào cũng tốt.

Thứ hai, mọi loại vi-rút máy tính đều phải chứa một chương trình để tự sao chép vào khu vực mà chương trình tìm kiếm định vị. Chương trình sao chép sẽ chỉ đủ tinh vi để thực hiện công việc của mình mà không bị phát hiện. Càng nhỏ càng tốt. Mức độ nhỏ của chương trình sẽ phụ thuộc vào mức độ phức tạp của vi-rút mà nó phải sao chép. Ví dụ, một loại vi-rút chỉ lây nhiễm các tệp COM có thể hoạt động với một chương trình sao chép nhỏ hơn nhiều so với một loại vi-rút lây nhiễm các tệp EXE. Điều này là do cấu trúc tệp EXE phức tạp hơn nhiều, vì vậy vi-rút chỉ cần làm nhiều hơn để tự gắn vào tệp EXE.

Trong khi vi-rút chỉ cần có khả năng định v các máy chủ phù hợp và tự gắn vào chúng, thì thường hữu ích khi kết hợp một số tính năng bổ sung vào vi-rút để tránh bị phát hiện, bởi người dùng máy tính hoặc bởi phần mềm phát hiện vi-rút thương mại. Các thói quen chống phát hiện có thể là một phần của các thói quen tìm kiếm hoặc sao chép hoặc tách biệt về mặt chức năng với chúng. Ví dụ, lối sống tìm kiếm có thể bị giới hạn nghiêm ngặt để tránh bị phát hiện. Một thói quen kiểm tra mọi tệp trên mọi ổ đĩa, không giới hạn, sẽ mất nhiều thời gian và gây ra đủ hoạt động đĩa bất thường mà người dùng cảnh giác có thể nghi ngờ. Ngoài ra, một thói quen chống phát hiện có thể khiến vi-rút kích hoạt trong một số điều kiện đặc biệt. Ví dụ, nó có thể chỉ kích hoạt sau khi một ngày nhất định đã trôi qua (do đó, vi-rút có thể nằm im trong một thời gian).

**Hình 1: Sơ đồ chức năng của vi-rút.**

Ngoài ra, nó chỉ có thể kích hoạt nếu một phím nào đó không được nhấn trong năm phút (cho thấy rằng người dùng không ở đó để xem máy tính của mình).

Các thói quen tìm kiếm, sao chép và chống phát hiện là những thành phần duy nhất cần thiết của vi-rút máy tính và chúng là những thành phần mà chúng ta sẽ tập trung vào trong tập này. Tất nhiên, nhiều vi-rút máy tính có các thói quen khác được thêm vào ngoài ba thói quen cơ bản để ngăn chặn hoạt động bình thường của máy tính, gây ra sự phá hủy hoặc để chơi khăm. Những thói quen như vậy có thể mang lại đặc điểm cho vi-rút, nhưng chúng không phải là yếu tố cần thiết cho sự tồn tại của vi-rút. Trên thực tế, những thói quen như vậy thường rất có hại cho mục tiêu sinh tồn và tự sinh sản của vi-rút, vì chúng khiến mọi người đều biết đến sự tồn tại của vi-rút. Nếu chỉ có một chút hoạt động của đĩa hơn dự kiến, có lẽ sẽ không ai nhận ra và vi-rút sẽ tiếp tục theo cách vui vẻ của nó. Mặt khác, nếu màn hình của chương trình yêu thích của một người nào đó hiện lên dòng chữ "Ha! Gotcha!" và sau đó toàn bộ máy tính bị khóa, mọi thứ trên đó bị hỏng, hầu hết mọi người đều có thể nhận ra rằng họ đã trở thành nạn nhân của một chương trình phá hoại. Và nếu họ thông minh, họ sẽ nhờ đến sự trợ giúp của chuyên gia để loại bỏ ngay lập tức. Kết quả là các loại vi-rút trên hệ thống cụ thể đó bị tiêu diệt, hoặc là do chính chúng hoặc do đội dọn dẹp.

Mặc dù có thể có trường hợp bất kỳ thứ gì không cần thiết cho sự sống còn của vi-rút có thể gây hại, nhiều vi-rút máy tính được viết chủ yếu để trở thành hệ thống phân phối thông minh của những “thói quen khác” này. Tác giả không quan tâm đến việc vi-rút có bị tiêu diệt trong khi hành động khi quả bom logic của nó phát nổ hay không, miễn là quả bom được triển khai hiệu quả. Sau đó, vi-rút trở nên giống như một phi công Kamikaze, người hy sinh “thói quen khác” này đã được chứng minh là khá sáng tạo. Ví dụ, một loại vi-rút nổi tiếng biến máy tính thành mô phỏng của một máy giặt, hoàn chỉnh với đồ họa và âm thanh. Một loại khác biến thứ Sáu ngày 13 thực sự thành một ngày tồi tệ bằng cách chỉ sống lại vào ngày đó và phá hủy dữ liệu. Tuy nhiên, những loại thói quen này thực sự là chủ đề của tập ba trong loạt bài này, tập thảo luận về các ứng dụng quân sự của vi-rút máy tính. Trong tập này, chúng ta sẽ tập trung vào những điều cơ bản về thiết kế hệ thống sinh sản. Và nếu bạn thực sự quan tâm đến các ứng dụng quân sự, hãy hãy nhớ rằng quả bom logic tốt nhất trên thế giới sẽ vô dụng nếu bạn không thể triển khai nó một cách chính xác. Hệ thống phân phối rất, rất quan trọng. Tình huống này tương tự như việc có một quả bom nguyên tử, nhưng không phải là phương tiện để đưa nó đi nửa vòng trái đất trong mười lăm phút. Chắc chắn, bạn có thể triển khai nó, nhưng việc vượt biên giới, đến gần mục tiêu, và giấu quả bom đều gây ra những rủi ro đáng kể. Nỗ lực để phát triển một tên lửa là xứng đáng.

**Các công cụ cần thiết để viết Virus**

Virus được viết bằng ngôn ngữ lắp ráp. Các ngôn ngữ cấp cao như Basic, C và Pascal được thiết kế để tạo ra các chương trình độc lập, nhưng các giả định do các ngôn ngữ này đưa ra khiến chúng gần như vô dụng khi viết virus. Chúng hoàn toàn không có khả năng thực hiện các động tác nhào lộn cần thiết để virus nhảy từ chương trình máy chủ này sang chương trình máy chủ khác. Điều đó không có nghĩa là người ta không thể thiết kế một ngôn ngữ cấp cao có thể thực hiện được công việc đó, nhưng chưa ai làm được như vậy. Do đó, để tạo ra virus, chúng ta phải sử dụng ngôn ngữ lắp ráp. Đây chỉ là cách duy nhất chúng ta có thể kiểm soát chặt chẽ tất cả các tài nguyên của hệ thống máy tính và sử dụng chúng theo cách chúng ta muốn, thay vì theo cách người khác nghĩ chúng ta nên làm.

Nếu bạn chưa từng lập trình bằng assembler trước đây, Tôi khuyên bạn nên tìm một hướng dẫn hay về chủ đề này để sử dụng cùng với cuốn sách này. (Một số được đề cập trong mục Đọc thêm ở cuối sách.) Trong các chương sau, tôi sẽ cho rằng kiến thức của bạn về các chi tiết kỹ thuật của PC—như cấu trúc tệp, lệnh gọi hàm, phân đoạn và thiết kế phần cứng—là hạn chế, và tôi sẽ cố gắng giải thích những vấn đề như vậy một cách cẩn thận ngay từ đầu. Tuy nhiên, tôi sẽ cho rằng bạn có một số kiến ​​thức về ngôn ngữ assembly ít nhất là ở mức độ mà bạn có thể hiểu được một số lệnh máy cơ bản, như mov ax,bx thực hiện. Nếu bạn không quen thuộc với lập trình ngôn ngữ assembly đơn giản hơn như thế này, hãy tìm một cuốn sách hướng dẫn về chủ đề này. Chỉ cần bỏ chút công sức, bạn sẽ bắt kịp.

Hiện nay, có ba trình biên dịch phổ biến trên thị trường và bạn sẽ cần một trong số chúng để thực hiện bất kỳ công việc nào với vi-rút máy tính. Trình biên dịch đầu tiên và lâu đời nhất là Macro Assembler của Microsoft, hay MASM viết tắt. Bạn sẽ mất khoảng 100 đô la để mua qua cửa hàng đặt hàng qua thư. Trình biên dịch thứ hai là Turbo Assembler của Borland, còn được gọi là TASM. Giá của nó cũng khoảng 100 đô la. Thứ ba, có A86, là phần mềm chia sẻ và có sẵn trên nhiều hệ thống bảng tin rên toàn quốc. Bạn có thể nhận được một bản sao miễn phí bằng cách gọi đến một trong những hệ thống này và tải xuống máy tính của bạn bằng một modem. Ngoài ra, một số công ty phần mềm cung cấp với giá khoảng 5 đô la qua thư. Tuy nhiên, nếu bạn có ý định sử dụng A86, tác giả yêu cầu bạn phải trả cho anh ta số tiền gần bằng khi bạn mua một trong những trình biên dịch khác. Anh ta sẽ buộc bạn phải chịu trách nhiệm về vi phạm bản quyền nếu anh ta có thể bắt được bạn. Cá nhân tôi không nghĩ A86 đáng giá. Tôi thích nhất là TASM, vì nó thực hiện chính xác những gì bạn yêu cầu mà không cố gắng qua mặt bạn. Đó chính xác là những gì bạn muốn khi viết một loại vi-rút. Bất kỳ điều gì ít hơn có thể khiến các chương trình của bạn có lỗi ngay cả khi chúng được viết đúng. Tuy nhiên, bất kỳ trình biên dịch nào bạn quyết định sử dụng, thì cả ba trình biên dịch vi-rút trong cuốn sách này đều có thể được biên dịch. Các tệp hàng loạt được cung cấp để thực hiện một trình biên dịch đúng với mỗi trình biên dịch.

Nếu bạn không có trình biên dịch, hoặc không có nguồn lực để mua một trình biên dịch, hoặc không có khuynh hướng học ngôn ngữ lắp ráp, thì các loại vi-rút được cung cấp ở định dạng hex của Intel để có thể tải trực tiếp vào máy tính của bạn dưới dạng thực thi. Đĩa các phiên bản đã biên dịch, có thể thực thi trực tiếp của từng loại vi-rút. Tuy nhiên, nếu bạn không hiểu mã nguồn ngôn ngữ lắp ráp, vui lòng không lấy các chương trình này và chạy chúng. Bạn chỉ đang tự chuốc lấy rắc rối, giống như một đứa trẻ bốn tuổi cầm một khẩu súng đã lên đạn.

**Trình lây nhiễm tệp COM đơn giản**

Trong chương này chúng ta sẽ thảo luận về một trong những cách đơn giản nhất virus máy tính. Loại virus này rất nhỏ, chỉ bao gồm 264 byte lệnh của ngôn ngữ máy. Nó cũng khá an toàn vì nó có một trong những quy trình tìm kiếm đơn giản nhất có thể. Vi-rút này, mà chúng tôi sẽ gọi là TIMID, được thiết kế để chỉ lây nhiễm các tệp COM nằm trong thư mục hiện được ghi trên máy tính. Nó có không nhảy qua các thư mục hoặc ổ đĩa nếu bạn không gọi nó từ thư mục khác để có thể dễ dàng chứa nó. Nó cũng vô hại bởi vì nó không chứa mã phá hoại và nó cho bạn biết khi nào nó lây nhiễm một tập tin mới, vì vậy bạn sẽ biết nó ở đâu và nó có ở đâu đi mất. Mặt khác, tính đơn giản cực độ của nó có nghĩa là đây là không phải là một loại virus rất hiệu quả. Nó sẽ không lây nhiễm vào hầu hết các tập tin và nó có thể dễ dàng bị bắt. Tuy nhiên, loại virus này sẽ giới thiệu tất cả những điều cần thiết. Các khái niệm cần thiết để viết một virus với độ phức tạp tối thiểu và rủi ro tối thiểu cho người thực nghiệm. Như vậy, nó là một điều tuyệt vời công cụ giảng dạy.

**Một số kiến ​​thức cơ bản về DOS**

Để hiểu cách thức virus tự sao chép từ chương trình này sang chương trình khác, chúng ta phải tìm hiểu chi tiết về cách. Hệ điều hành DOS tải chương trình vào bộ nhớ và chuyển quyền điều khiển cho nó. Virus phải được thiết kế sao cho mã của nó có thể được thực thi, thay vì chỉ chương trình mà nó gắn vào. Chỉ một thì nó có thể sinh sản được không. Sau đó, nó phải có khả năng chuyển quyền điều khiển trở lại chương trình máy chủ, do đó máy chủ cũng có thể thực thi toàn bộ chương trình đó.

Khi nhập tên chương trình tại dấu nhắc DOS, DOS bắt đầu tìm kiếm các tập tin có tên đó và phạm vi “COM”. Nếu tìm thấy nó sẽ tải tập tin vào bộ nhớ và thực thi Nó. Nếu không DOS sẽ tìm kiếm các tập tin có cùng tên và phạm vi của “EXE” để tải và thực thi. Nếu không tìm thấy tệp EXE, hệ điều hành cuối cùng sẽ tìm kiếm một tệp có phạm vi “BAT” để thực thi. Thất bại cả ba khả năng này, DOS sẽ hiển thị thông báo lỗi “Lệnh hoặc tên tệp không hợp lệ”.

Các tệp EXE và COM được Trung tâm thực thi trực tiếp đơn vị xử lý. Trong hai loại file chương trình này, file COM đơn giản hơn nhiều. Chúng có định dạng phân đoạn được xác định trước được tích hợp sẵn trong cấu trúc của DOS, trong khi các tệp EXE được thiết kế để xử lý định dạng phân đoạn do người dùng xác định, điển hình là rất lớn và các chương trình phức tạp. Tệp COM là hình ảnh nhị phân trực tiếp của những gì cần được đưa vào bộ nhớ và được CPU thực thi, nhưng một Tệp EXE thì không.

Để thực thi một tập tin COM, DOS phải thực hiện một số bước chuẩn bị làm việc trước khi trao quyền kiểm soát chương trình đó. Quan trọng nhất là DOS kiểm soát và phân bổ việc sử dụng bộ nhớ trong máy tính. Vì thế đầu tiên nó kiểm tra xem có đủ chỗ trong bộ nhớ để tải chương trình. Nếu có thể, DOS sẽ phân bổ bộ nhớ cần thiết cho chương trình. Bước này không khác gì một công việc dọn phòng nội bộ chức năng. DOS chỉ ghi lại dung lượng trống mà nó dành cho chương trình này và chương trình khác, do đó nó sẽ không cố tải một chương trình khác. chương trình lên trên nó sau hoặc cung cấp không gian bộ nhớ cho chương trình điều đó sẽ xung đột với một chương trình khác. Bước đi như vậy là cần thiết bởi vì nhiều chương trình có thể nằm trong bộ nhớ tại bất kỳ thời điểm nào thời gian. Ví dụ: các chương trình bật lên, lưu trữ trong bộ nhớ có thể vẫn còn trong bộ nhớ và chương trình cha có thể tải chương trình con vào bộ nhớ, thực thi và sau đó trả lại quyền điều khiển cho cha mẹ.

Tiếp theo, DOS xây dựng một khối bộ nhớ dài 256 byte

được gọi là Tiền tố phân đoạn chương trình hoặc PSP. PSP là một

tàn dư của hệ điều hành cũ hơn được gọi là CP/M. CP/M là

phổ biến vào cuối những năm bảy mươi và đầu những năm tám mươi như một hoạt động

hệ thống dành cho máy vi tính dựa trên bộ vi xử lý 8080 và Z80. Trong thế giới CP/M, 64 kilobyte là toàn bộ bộ nhớ

máy tính đã có. 256 byte thấp nhất của bộ nhớ đó đã được dành riêng

để hệ điều hành tự lưu trữ dữ liệu quan trọng. Ví dụ,

vị trí 5 trong bộ nhớ chứa lệnh nhảy để đến phần còn lại

của hệ điều hành, được lưu trữ trong bộ nhớ cao và

vị trí khác nhau tùy theo dung lượng bộ nhớ của máy tính

có. Do đó, các chương trình được viết cho các máy này sẽ truy cập vào

hệ điều hành hoạt động bằng cách gọi vị trí 5 trong bộ nhớ. Khi

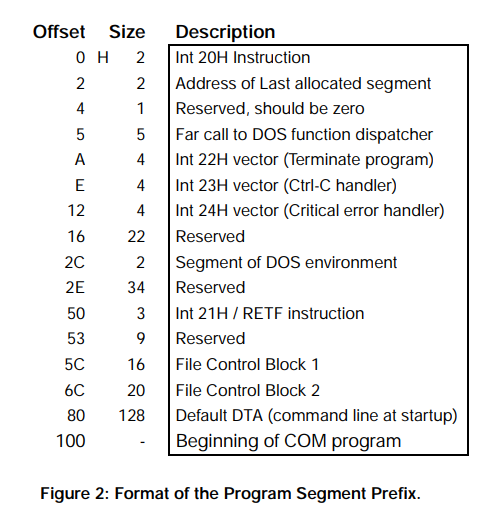
PC-DOS xuất hiện, nó bắt chước CP/M vì CP/M rất

phổ biến và nhiều chương trình đã được viết để làm việc với nó. Vì thế

PSP (và toàn bộ khái niệm tệp COM) đã trở thành một phần của DOS. các

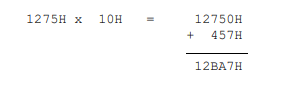
kết quả là rất nhiều thông tin được lưu trữ trong PSP ngày nay ít được sử dụng bởi một lập trình viên DOS. Tuy nhiên, một số thông tin đó rất hữu ích vì chúng tôi

lát nữa sẽ thấy.



Khi PSP được xây dựng, DOS sẽ lấy tệp COM được lưu trữ trên đĩa và tải nó vào bộ nhớ ngay phía trên PSP, bắt đầu từ offset 100H. Một khi điều này được thực hiện, DOS gần như sẵn sàng chuyển quyền điều khiển cho chương trình. Tuy nhiên, trước khi thực hiện, nó phải thiết lập các thanh ghi trong CPU tới các giá trị được xác định trước nhất định. Đầu tiên, đoạn đăng ký phải được đặt đúng cách, nếu không chương trình COM không thể chạy. Chúng ta hãy lấy một hãy xem cách thức và lý do của các thanh ghi phân đoạn này.

Trong bộ vi xử lý 8088, tất cả các thanh ghi đều là thanh ghi 16 bit. Vấn đề là thanh ghi 16 bit sẽ chỉ cho phép một địa chỉ 64 kilobyte bộ nhớ. Nếu bạn muốn sử dụng nhiều bộ nhớ hơn, bạn cần nhiều bit hơn để giải quyết nó. 8088 có thể đánh địa chỉ tới một megabyte bộ nhớ bằng cách sử dụng một quá trình được gọi là phân đoạn. Nó sử dụng hai thanh ghi để tạo địa chỉ bộ nhớ vật lý là 20 dài bit thay vì chỉ 16. Cặp thanh ghi như vậy bao gồm một đoạn thanh ghi chứa các bit quan trọng nhất của địa chỉ và một thanh ghi bù, chứa các bit có ý nghĩa nhỏ nhất. các thanh ghi phân đoạn trỏ đến khối bộ nhớ 16 byte và phần bùthanh ghi cho biết cần thêm bao nhiêu byte vào đầu khối 16 byte để xác định vị trí byte mong muốn trong bộ nhớ. Ví dụ, nếu thanh ghi ds được đặt thành 1275 Hex và thanh ghi bx được đặt thành 457 Hex, sau đó địa chỉ vật lý 20 bit của byte ds:[bx] là

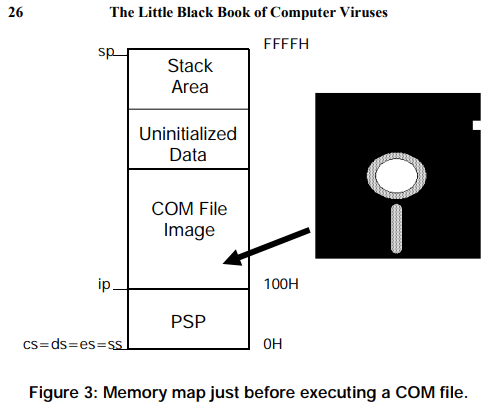


Không có phần bù nào phải lớn hơn 15, nhưng một thường sử dụng các giá trị lên tới toàn bộ phạm vi 64 kilobyte của phần bù đăng ký. Điều này dẫn đến khả năng viết một tập tin vật lý duy nhất địa chỉ theo nhiều cách khác nhau. Ví dụ đặt ds=12BA Hex và bx = 7 sẽ tạo ra cùng một địa chỉ vật lý 12BA7 Hex như trong ví dụ trên. Sự lựa chọn đúng đắn đơn giản là bất cứ điều gì thuận tiện cho người lập trình. Tuy nhiên, thông lệ lập trình tiêu chuẩn là thiết lập các thanh ghi phân đoạn và để chúng nguyên như càng nhiều càng tốt, sử dụng offset để phân loại càng nhiều dữ liệu và mã như người ta có thể (64 kilobyte nếu cần thiết).

8088 có 4 thanh ghi đoạn cs, ds, ss và es, viết tắt của Phân đoạn mã, Phân đoạn dữ liệu, Phân đoạn ngăn xếp và Phân khúc bổ sung tương ứng. Mỗi người phục vụ các mục đích khác nhau. Thanh ghi cs chỉ định phân đoạn 64K nơi chương trình thực tế các lệnh được thực thi bởi CPU được định vị. Dữ liệuPhân đoạn được sử dụng để chỉ định một phân đoạn để đưa dữ liệu của chương trình vào, và Phân đoạn ngăn xếp chỉ định vị trí ngăn xếp của chương trình xác định vị trí. Thanh ghi es có sẵn dưới dạng thanh ghi phân đoạn bổ sung cho việc sử dụng của lập trình viên. Nó thường có thể được sử dụng để trỏ đến đoạn bộ nhớ video, để ghi dữ liệu trực tiếp vào video, v.v.

Các tập tin COM được thiết kế để hoạt động với một cách rất đơn giản nhưng cấu trúc phân đoạn giới hạn. cụ thể là họ có một phân khúc, cs=ds=es=ss. Tất cả dữ liệu được lưu trữ trong cùng phân đoạn với chương trình chính mã đó và ngăn xếp chia sẻ phân đoạn này. Vì bất kỳ điều gì đã cho đoạn dài 64 kilobyte, một chương trình COM có thể sử dụng tối đa 64 kilobyte cho tất cả mã, dữ liệu và ngăn xếp của nó. Khi PC lần đầu tiên xuất hiện được giới thiệu, mọi người đã quen với việc viết các chương trình giới hạn ở 64 kilobyte và có vẻ như rất nhiều bộ nhớ. Tuy nhiên, ngày nay nó không có gì lạ khi tìm thấy các chương trình yêu cầu hàng trăm kilobyte mã và có thể nhiều dữ liệu. Những chương trình như vậy phải sử dụng sơ đồ phân đoạn phức tạp hơn định dạng tệp COM cho phép. Cấu trúc tệp EXE được thiết kế để xử lý sự phức tạp đó. Hạn chế của tệp EXE là mã chương trìnhđược lưu trữ trên đĩa phải được sửa đổi đáng kể trước khi có thể được thực hiện bởi CPU. DOS thực hiện điều đó vào lúc tải và đó là hoàn toàn trong suốt đối với người dùng, nhưng có một loại virus gắn vào EXE các tập tin không được làm phiền DOS trong quá trình sửa đổi này, nếu không nó sẽ sẽ không hoạt động. Chương trình COM không yêu cầu sửa đổi này xử lý vì nó chỉ sử dụng một phân đoạn cho mọi thứ. Cái này cho phép lưu trữ hình ảnh nhị phân thẳng của mã được thực thi trên đĩa (tệp COM). Khi đến lúc chạy chương trình, DOS chỉ cần thiết lập các thanh ghi phân đoạn đúng cách và thực hiện nó.

PSP được thiết lập ở đầu phân đoạn được phân bổ đối với tệp COM, tức là ở độ lệch 0. DOS chọn phân đoạn dựa trên bộ nhớ trống nào còn trống và đặt PSP ngay từ đầu phân khúc đó. Bản thân tệp COM được tải ở offset 100 Hex, chỉ sau PSP. Khi mọi thứ đã sẵn sàng, DOS chuyển quyền điều khiển về phần đầu của chương trình bằng cách nhảy tới phần bù 100 Hex trong đoạn mã nơi chương trình được tải. Từ đó trở đi chương trình chạy và thỉnh thoảng nó truy cập vào DOS khi thấy phù hợp, để thực hiện các chức năng I/O khác nhau, như đọc và ghi vào đĩa. Khi chương trình hoàn tất, nó chuyển quyền điều khiển trở lại DOS và DOS giải phóng bộ nhớ dành riêng cho chương trình đó và cung cấp cho người dùng một dấu nhắc dòng lệnh khác.

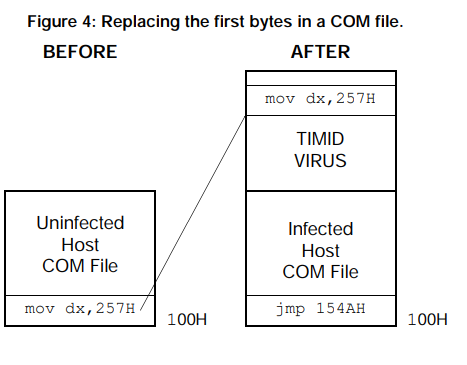


**Sơ lược về virus**

Để vi-rút cư trú trong tệp COM, nó phải có điều khiển được chuyển đến mã của nó tại một thời điểm nào đó trong quá trình thực thi lệnh chương trình. Có thể hình dung virus có thể kiểm tra file COM và xác định xem nó có thể giành quyền kiểm soát từ chương trình như thế nào vào bất kỳ lúc nào. điểm trong quá trình thực hiện nó. Việc phân tích như vậy sẽ rất khó khăn, tuy nhiên, đối với trường hợp chung, virus tạo ra sẽ là bất cứ điều gì nhưng đơn giản. Cho đến nay, điểm dễ dàng nhất để kiểm soát là đúng ngay từ đầu, khi DOS nhảy đến phần bắt đầu của chương trình. Lúc này, virus hoàn toàn có quyền sử dụng bất kỳ khoảng trống nào phía trên chương trình. hình ảnh của tệp COM được DOS tải vào bộ nhớ.Vì bản thân chương trình chưa được thực thi nên nó không thể thiết lập dữ liệu ở bất cứ đâu trong bộ nhớ hoặc di chuyển ngăn xếp, vì vậy đây là cách rất an toàn thời gian để virus hoạt động. Ở giai đoạn này, đó không phải là một nhiệm vụ quá khó khăn để đảm bảo rằng virus sẽ không can thiệp vào chương trình máy chủlàm hỏng nó hoặc làm cho nó không hoạt động. Khi chương trình chủ trì bắt đầu. Tuy nhiên, để thực thi thì hầu hết mọi điều đều có thể xảy ra và công việc của virus trở nên khó khăn hơn nhiều.

Để giành quyền kiểm soát khi khởi động, vi-rút lây nhiễm COM tệp phải thay thế một vài byte đầu tiên trong tệp COM bằng bước nhảy tớimã virus, có thể được thêm vào cuối tệp COM. Sau đó, khi tập tin COM được thực thi, nó sẽ nhảy vào virus, virus này tiếp tục tìm kiếm nhiều tập tin hơn để lây nhiễm và lây nhiễm chúng.Khi virus đã sẵn sàng, nó có thể trả lại quyền điều khiển cho chương trình chủ. Vấn đề khi thực hiện việc này là virus đã thay thế phần mềm đầu tiên vài byte của chương trình chủ với mã riêng của nó. Như vậy nó phải khôi phục các byte đó và sau đó quay lại bù 100 Hex, trong đó chương trình ban đầu bắt đầu.

Sau đây là kế hoạch cơ bản cho việc lây nhiễm virus đơn giản một tập tin COM. Hãy tưởng tượng một con virus đang nằm trong bộ nhớ vừa được kích hoạt. Nó đi ra ngoài và lây nhiễm vào một tệp COM khác. Bước chân từng bước một, nó có thể hoạt động như thế này:



1. Tệp COM bị nhiễm được tải vào bộ nhớ và bị xử tử. Mã virus được kiểm soát đầu tiên.

2. Virus trong bộ nhớ sẽ tìm kiếm trên đĩa để tìm một tập tin COM phù hợp để lây nhiễm.

3. Nếu tìm thấy một tệp phù hợp, vi-rút sẽ thêm vào tệp riêng của nó mã ở cuối tập tin.

4. Tiếp theo, nó đọc vài byte đầu tiên của tệp vào bộ nhớ và ghi chúng trở lại tập tin trong một vùng dữ liệu đặc biệt trong mã của virus. Loại virus mới sẽ cần những byte này khi nó thực thi.

5. Tiếp theo, virus trong bộ nhớ ghi lệnh nhảy tới phần đầu của tập tin bị lây nhiễm, nó sẽ vượt qua kiểm soát virus mới khi chương trình máy chủ của nó bị xử tử.

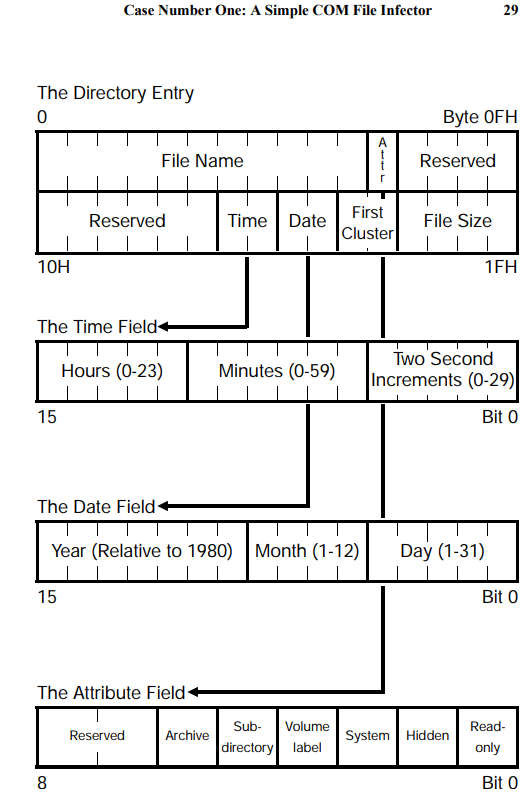
6. Sau đó, virus trong bộ nhớ sẽ lấy các byte đã được ban đầu là các byte đầu tiên trong máy chủ của nó và đặt chúng trở lại (ở độ lệch 100H).

7. Cuối cùng, mã virus nhảy tới offset 100 Hex và cho phép chương trình máy chủ của nó thực thi.

Được rồi. Vì vậy, hãy phát triển một loại virus thực sự với các thông số kỹ thuật này. chúng tôi sẽ cần cả cơ chế tìm kiếm và cơ chế sao chép.

**Cơ chế tìm kiếm**

Để hiểu cách virus tìm kiếm các tập tin mới để lây nhiễm trên một máy tính kiểu IBM PC hoạt động với MS-DOS hoặc PCDOS, điều quan trọng là phải hiểu cách DOS lưu trữ các tập tin và thông tin về họ. Tất cả thông tin về mọi tập tin trên đĩa được lưu trữ trong hai khu vực trên đĩa, được gọi là thư mục và Bảng phân bổ tệp hay viết tắt là FAT. Thư mục chứa 32 bản ghi mô tả tệp byte cho mỗi tệp. Bản ghi mô tả này chứa tên và phạm vi của tệp, kích thước, ngày và giờ của tệp việc tạo và thuộc tính tệp chứa thông tin cần thiết cho hệ điều hành về cách xử lý tệp. FAT là một bản đồ của toàn bộ đĩa, chỉ đơn giản là thông báo cho hệ điều hành khu vực nào bị chiếm bởi tập tin nào.



Mỗi đĩa có hai FAT, là các bản sao giống hệt nhau của mỗi đĩa. khác. Cái thứ hai là bản sao lưu, phòng trường hợp cái thứ nhất bị hỏng. TRÊN mặt khác, một đĩa có thể có nhiều thư mục. Một thư mục, được gọi là thư mục gốc, có mặt trên mọi đĩa, nhưng thư mục gốc có thể có nhiều thư mục con, lồng vào nhau để tạo thành một cấu trúc cây. Các thư mục con này có thể được tạo, sử dụng và người dùng tùy ý loại bỏ. Vì vậy, cấu trúc cây có thể đơn giản như hoặc phức tạp như người dùng đã thực hiện.

Cả FAT và thư mục gốc đều được đặt ở một vị trí cố định khu vực của đĩa, dành riêng cho họ. Thư mục con là được lưu trữ giống như các tệp khác có thuộc tính tệp được đặt để chỉ ra rằng tập tin này là một thư mục. Hệ điều hành sau đó xử lý việc này tệp thư mục con theo cách hoàn toàn khác với các tệp khác để làm cho nó trông giống như một thư mục chứ không chỉ là một tập tin khác. các tệp thư mục con chỉ đơn giản bao gồm một chuỗi các bản ghi 32 byte mô tả các tập tin trong thư mục đó. Nó có thể chứa bản ghi 32 byte với thuộc tính được đặt thành thư mục, có nghĩa là tệp này là một thư mục con của một thư mục con.

Hệ điều hành DOS thường kiểm soát mọi quyền truy cập vào tập tin và thư mục con. Nếu một người muốn đọc hoặc ghi vào một tập tin, anh ta không viết chương trình định vị đúng thư mục trên đĩa, đọc các bản ghi mô tả tập tin để tìm đúng bản ghi, hình ra nơi tập tin và đọc nó. Thay vì làm tất cả công việc này, anh ta chỉ cần đưa cho DOS thư mục và tên của tập tin rồi hỏi nó để mở tập tin. DOS thực hiện tất cả công việc khó khăn. Điều này tiết kiệm rất nhiều thời gian viết và gỡ lỗi chương trình. Đơn giản là người ta không có để giải quyết các chi tiết phức tạp của việc quản lý tập tin và giao tiếp với phần cứng.

DOS được yêu cầu phải làm gì bằng cách sử dụng các chương trình dịch vụ ngắt (ISR). Ngắt 21H là chương trình phục vụ ngắt chính của DOS mà chúng tôi sẽ sử dụng. Để gọi ISR, người ta chỉ cần thiết lập yêu cầu CPU đăng ký với bất kỳ giá trị nào mà ISR cần để biết phải làm gì làm và gọi ngắt. Ví dụ: mã mở một tệp có tên được lưu trong vị trí bộ nhớ FNAME chuẩn bị cho việc đọc nó vào bộ nhớ. Hàm này báo cho DOS để xác định vị trí tập tin và chuẩn bị cho việc đọc. Lệnh “int 21H” chuyển quyền điều khiển sang DOS và để nó thực hiện công việc của mình. Khi DOS là mở file xong, điều khiển quay lại câu lệnh ngay sau “int 21H”. Thanh ghi ah chứa hàm số mà DOS sử dụng để xác định xem bạn đang yêu cầu nó làm gì. Các thanh ghi khác phải được thiết lập khác nhau, tùy thuộc vào những gì à, là để truyền tải thêm thông tin tới DOS về những gì nó được cho là để làm. Trong ví dụ trên, cặp thanh ghi ds:dx được sử dụng để trỏ đến vị trí bộ nhớ nơi lưu trữ tên của tệp cần mở. Thanh ghi al báo cho DOS mở file chỉ để đọc

Tất cả các chức năng DOS khác nhau, bao gồm cả cách thiết lập tất cả các sổ đăng ký đều được trình bày chi tiết trong nhiều cuốn sách về chủ đề này. Peter Hướng dẫn lập trình viên cho máy tính IBM của Norton là một trong những tài liệu hay hơn những cái đó, vì vậy nếu bạn không có sẵn thông tin đó, tôi đề nghị bạn lấy một bản sao. Ở đây chúng ta sẽ chỉ thảo luận về DOS các chức năng chúng ta cần, khi chúng ta cần chúng. Điều này có lẽ sẽ đủ để vượt qua. Tuy nhiên, nếu bạn định viết virus cho riêng mình, chắc chắn đáng để biết về tất cả các chức năng khác nhau mà bạn có thể sử dụng, cũng như các chi tiết chi tiết hơn về cách chúng hoạt động và những gì cần chú ý.

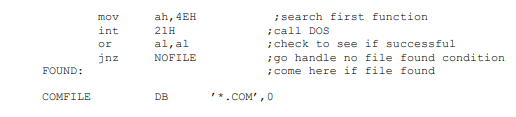
Để viết một chương trình tìm kiếm các tập tin khác có thể bị lây nhiễm, chúng ta sẽ sử dụng các chức năng tìm kiếm của DOS. Những người đã viết DOS biết rằng nhiều chương trình (không chỉ virus) yêu cầu khả năng tìm kiếm các tập tin và thao tác trên chúng nếu có bất kỳ loại được yêu cầu nào thành lập. Vì vậy, họ đã kết hợp một cặp chức năng tìm kiếm vào trình xử lý ngắt 21H, được gọi là Tìm kiếm đầu tiên và Tìm kiếm tiếp theo. Đây là một số hàm DOS phức tạp hơn, vì vậy chúng yêu cầu người dùng phải thực hiện khá nhiều công việc chuẩn bị trước khi gọi họ. Bước đầu tiên là thiết lập chuỗi ASCIIZ trong bộ nhớ để chỉ định thư mục cần tìm và những tập tin cần tìm. Cái này chỉ đơn giản là một mảng byte được kết thúc bằng một byte rỗng (0). DOS có thể Trường hợp thứ nhất: Kẻ lây nhiễm tệp COM đơn giản 31 tìm kiếm và báo cáo về tất cả các tệp trong một thư mục hoặc một tập hợp con của các tệp mà người dùng có thể chỉ định theo thuộc tính tệp và bằng cách chỉ định tên tệp sử dụng ký tự đại diện “?” và “\*”, mà bạn nên làm quen với việc thực thi các lệnh như copy \*.\* a: và dir a???\_100.\* từ dòng lệnh trong DOS. (Nếu không, cơ bản sách về DOS sẽ giải thích cú pháp này.) Ví dụ: ASCIIZ string sẽ thiết lập chức năng tìm kiếm để tìm kiếm tất cả các file có tên hyper và bất kỳ mức độ nào có thể, trong thư mục con có tên system. DOS có thể tìm thấy các tệp như hyper.c, hyper.prn, hyper.exe, v.v.

Sau khi thiết lập chuỗi ASCIIZ này, người ta phải đặt đăng ký ds và dx theo phân đoạn và offset của ASCIIZ này chuỗi trong bộ nhớ. Đăng ký cl phải được đặt thành mặt nạ thuộc tính tệp nó sẽ cho DOS biết những thuộc tính tệp nào được cho phép trong tìm kiếm và cái nào cần loại trừ. Logic đằng sau mặt nạ thuộc tính này có phần phức tạp nên bạn có thể muốn nghiên cứu chi tiết ở Phụ lục G. Cuối cùng, để gọi hàm Search First, người ta phải đặt ah = 4E Hex.

Nếu chức năng tìm kiếm đầu tiên thành công, nó sẽ trả về với đăng ký al = 0 và nó định dạng 43 byte dữ liệu trong Disk Transfer Khu vực hoặc DTA. Dữ liệu này cung cấp cho chương trình thực hiện tìm kiếm tên của tập tin mà DOS vừa tìm thấy, thuộc tính, kích thước và ngày tạo ra nó. Một số dữ liệu được báo cáo trong DTA cũng được DOS sử dụng để thực hiện chức năng Tìm kiếm tiếp theo. Nếu việc tìm kiếm không thể tìm thấy tệp phù hợp, DOS trả về giá trị khác 0, không có dữ liệu trong DTA. Vì chương trình gọi biết địa chỉ của DTA, nó có thể kiểm tra khu vực đó để tìm thông tin tệp sau DOS đã lưu trữ nó ở đó.

Để xem chức năng này hoạt động rõ ràng hơn như thế nào, chúng ta hãy xem xét một ví dụ Giả sử chúng ta muốn tìm tất cả các tệp trong thư mục hiện tại thư mục được ghi lại có phạm vi “COM”, bao gồm cả thư mục ẩn và tập tin hệ thống. Mã hợp ngữ để thực hiện Tìm kiếm đầu tiên sẽ trông như thế này (giả sử ds đã được thiết lập chính xác):



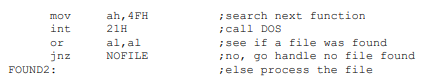


Nếu quy trình này được thực thi thành công, DTA có thể trông như thế này:



khi chương trình đạt đến nhãn FOUND. Trong trường hợp này việc tìm kiếm đã tìm thấy tệp COMMAND.COM.

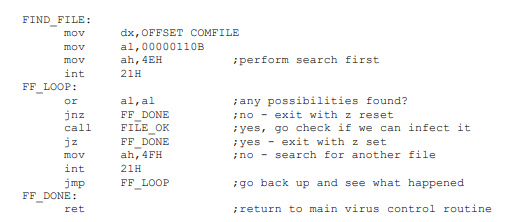
So với chức năng Tìm kiếm đầu tiên, chức năng Tìm kiếm Việc tiếp theo thật dễ dàng vì tất cả dữ liệu đã được thiết lập sẵn bởi Tìm kiếm đầu tiên. Chỉ cần đặt ah = 4F hex và gọi ngắt DOS 21H:

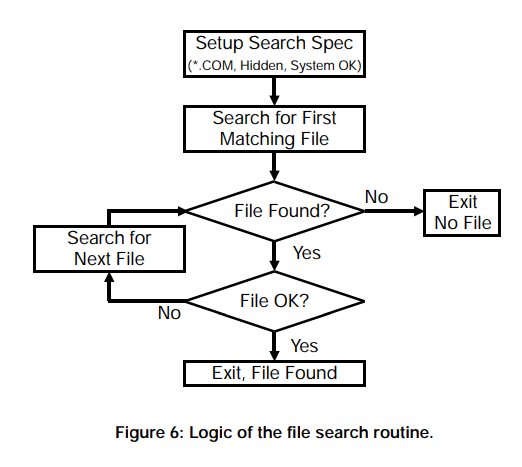


Nếu tìm thấy một tệp khác, dữ liệu trong DTA sẽ được cập nhật với tên tệp mới và ah sẽ được đặt thành 0 khi trả về. Nếu không còn nữa tìm thấy kết quả phù hợp, DOS sẽ đặt ah thành giá trị nào đó ngoài 0 trên trở lại. Ở đây người ta phải cẩn thận để dữ liệu trong DTA không bị thay đổi giữa lệnh gọi Tìm kiếm đầu tiên và lệnh gọi sau đó tới Tìm kiếm tiếp theo, bởi vì Tìm kiếm tiếp theo mong đợi dữ liệu từ lệnh gọi tìm kiếm cuối cùng ở đó.

Tất nhiên virus máy tính không cần phải tìm kiếm thông qua tất cả các tập tin COM trong một thư mục. Nó phải tìm một cái sẽ thích hợp để lây nhiễm, và sau đó lây nhiễm nó. Chúng ta hãy tưởng tượng một thủ tục FILE\_OK. Với tên của một tập tin trên đĩa, nó sẽ xác định xem tập tin đó có tốt để lây nhiễm hay không. Nếu nó có khả năng lây nhiễm, FILE\_OK sẽ trả về với cờ 0, z, được đặt, nếu không nó sẽ trở lại với việc đặt lại cờ 0. Chúng ta có thể sử dụng cờ này để xác định có nên tiếp tục tìm kiếm các tập tin khác hay không, hoặc liệu chúng ta có nên hãy lây nhiễm cái mà chúng ta đã tìm thấy.

Nếu toàn bộ cơ chế tìm kiếm của chúng tôi cũng sử dụng cờ z để báo cho chương trình điều khiển chính rằng nó đã tìm thấy một tập tin bị lây nhiễm (z=tìm thấy tệp, nz=không tìm thấy tệp) thì chức năng tìm kiếm đã hoàn thành của chúng tôi có thể được viết như thế này:





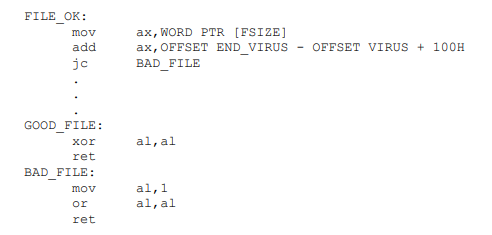
Nghiên cứu thói quen tìm kiếm này một cách cẩn thận. Điều quan trọng là phải hiểu nếu bạn muốn viết virus máy tính, và tổng quát hơn, nó rất hữu ích trong nhiều loại chương trình khác nhau.

ul trong rất nhiều chương trình thuộc mọi thể loại. Tất nhiên, để virus của chúng ta hoạt động chính xác, chúng ta phải viết hàm FILE\_OK xác định xem một tập tin có nên bị nhiễm bệnh hoặc bị bỏ lại một mình. Chức năng này đặc biệt quan trọng đối với sự thành công hay thất bại của virus, bởi vì nó cho virus biết khi nào và chuyển đi đâu. Nếu nó bảo virus lây nhiễm vào một chương trình có khả năng lây nhiễm không có chỗ cho virus thì chương trình mới bị nhiễm có thể vô tình bị hủy hoại. Hoặc nếu FILE\_OK không thể biết liệu chương trình đã bị nhiễm virus, nó sẽ ra lệnh cho virus tiếp tục và lây nhiễm cùng một tập tin hết lần này đến lần khác. Sau đó tập tin sẽ ngày càng lớn hơn cho đến khi không còn chỗ cho sự nhiễm trùng. Ví dụ, thủ tục chỉ đơn giản đặt cờ z và trả về. Nếu thói quen tìm kiếm của chúng tôi sử dụng điều này chương trình con, nó sẽ luôn dừng và nói rằng tệp COM đầu tiên nó được tìm thấy là người đã lây nhiễm. Kết quả sẽ là COM đầu tiên chương trình trong một thư mục sẽ là chương trình duy nhất có thể bị nhiễm. Nó sẽ tiếp tục bị nhiễm trùng lặp đi lặp lại, và ngày càng tăng kích thước cho đến khi nó vượt quá giới hạn kích thước và bị hỏng. Vì thế mặc dù ví dụ trên về FILE\_OK có thể cho phép virus tấn công lây nhiễm ít nhất một tệp, nó sẽ không hoạt động đủ tốt đối với vi-rút để có thể bắt đầu nhảy từ tập tin này sang tập tin khác.

Một quy trình FILE\_OK tốt phải thực hiện hai lần kiểm tra: (1) nó phải kiểm tra một tệp để xem nó có quá dài để đính kèm vi-rút không và (2) nó phải kiểm tra xem virus đã có ở đó chưa. Nếu tập tin là đủ ngắn và không có vi-rút, FILE\_OK sẽ quay trở lại “tiếp tục” với quy trình tìm kiếm.

Khi nhập vào FILE\_OK, chức năng tìm kiếm đã thiết lập DTA với 43 byte thông tin về file cần kiểm tra, bao gồm kích thước và tên của nó. Giả sử chúng ta đã xác định hai nhãn, FSIZE và FNAME trong DTA để truy cập kích thước tệp và tên tệp tương ứng. Sau đó kiểm tra kích thước file xem virus có vừa không là một vấn đề đơn giản Vì kích thước tệp của tệp COM luôn nhỏ hơn Trường hợp thứ nhất: Kẻ lây nhiễm tệp COM đơn giản 35 hơn 64 kilobyte, chúng tôi có thể tải kích thước của tệp mà chúng tôi muốn lây nhiễm vào thanh ghi rìu:

Tiếp theo chúng ta thêm số byte mà virus sẽ phải thêm vào vào tập tin này, cộng thêm 100H. 100H là cần thiết vì DOS cũng sẽ phân bổ chỗ cho PSP và tải tệp chương trình ở độ lệch 100H. Để tự động xác định số byte mà virus sẽ cần, chúng tôi chỉ cần đặt nhãn VIRUS ở đầu mã vi-rút mà chúng tôi đang có viết và nhãn END\_VIRUS ở cuối nó và lấy sự khác biệt. Nếu chúng ta thêm các byte này vào ax và ax tràn ra thì tập tin mà quá trình tìm kiếm đã tìm thấy là quá lớn để cho phép lây nhiễm thành công. Tràn sẽ làm cho cờ nhớ c bị được đặt, vì vậy quá trình kiểm tra kích thước tệp sẽ trông giống như thế này:



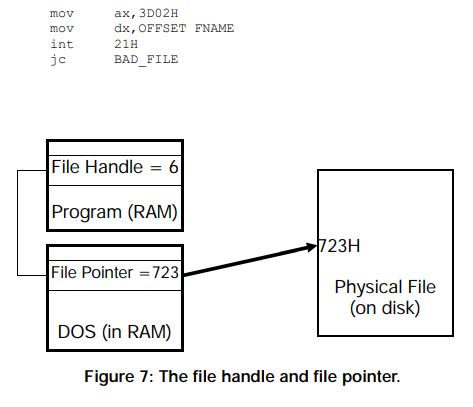
Quy trình này sẽ đủ để ngăn chặn vi-rút lây nhiễm vào bất kỳ tệp nào cái đó quá lớn.

Vấn đề tiếp theo mà thủ tục FILE\_OK phải giải quyết là cách tránh lây nhiễm vào một tập tin đã bị lây nhiễm. Cái này chỉ có thể được thực hiện nếu virus có một số hiểu biết về việc lây nhiễm một tập tin diễn ra như thế nào. Trong virus TIMID, chúng ta có quyết định thay thế vài byte đầu tiên của chương trình chủ bằng một nhảy tới mã virus. Do đó, thủ tục FILE\_OK có thể bị lỗi và đọc tệp có nguy cơ lây nhiễm để xác định lệnh đầu tiên của nó có phải là một bước nhảy hay không. Nếu không thì virus rõ ràng là chưa lây nhiễm tập tin đó. Có hai kiểu nhảy 36 Cuốn sách đen nhỏ về virus máy tính các hướng dẫn có thể gặp trong tệp COM, được gọi là nhảy gần và nhảy ngắn. Virus chúng tôi tạo ra ở đây sẽ luôn sử dụng bước nhảy gần để giành quyền kiểm soát khi chương trình bắt đầu. Vì một bước nhảy ngắn chỉ có phạm vi 128 byte, chúng tôi không thể sử dụng nó để lây nhiễm vào tệp COM lớn hơn 128 byte. Cú nhảy gần cho phép phạm vi 64 kilobyte. Vì vậy, nó luôn có thể được sử dụng để nhảy từ đầu file COM có virus, ở cuối chương trình không có bất kể tệp COM lớn đến mức nào (miễn là nó thực sự là một COM hợp lệ tài liệu). Bước nhảy gần được thể hiện bằng ngôn ngữ máy với byte E9 Hex, theo sau là hai byte cho CPU biết khoảng cách nhảy. Vì vậy, thử nghiệm đầu tiên của chúng tôi để xem liệu nhiễm trùng đã xảy ra hay chưa là kiểm tra để xem byte đầu tiên trong tệp có phải là E9 Hex hay không. Nếu đó là bất cứ điều gì khác, virus rõ ràng sẽ tiến lên và lây nhiễm.

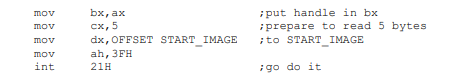
Tuy nhiên, tìm kiếm E9 Hex là chưa đủ. Nhiều tập tin COM được thiết kế sao cho lệnh đầu tiên là một bước nhảy để bắt đầu. Như vậy vi rút thậm chí có thể gặp các tệp bắt đầu bằng E9 Hex mặc dù họ chưa bao giờ bị nhiễm bệnh. Virus không thể cho rằng một tập tin đã bị nhiễm chỉ vì nó bắt đầu bằng E9. Nó phải đi xa hơn. Nó phải có cách để biết liệu một tập tin có bị nhiễm virus hay không ngay cả khi nó bắt đầu với E9. Nếu chúng ta không kết hợp thêm điều này bước vào quy trình FILE\_OK, vi-rút sẽ vượt qua nhiều điều tốt COM mà nó có thể lây nhiễm vì nó cho rằng chúng đã bị nhiễm bệnh. Mặc dù không kết hợp được tính năng như vậy vào FILE\_OK sẽ không khiến virus bị lỗi mà sẽ hạn chế chức năng của nó.

Một cách để làm cho bài kiểm tra này đơn giản nhưng vẫn rất đáng tin cậy là để thay đổi nhiều byte hơn mức cần thiết khi bắt đầu chương trình chủ. Bước nhảy gần sẽ cần ba byte, vì vậy chúng ta có thể lấy thêm hai cái nữa và mã hóa chúng theo một cách duy nhất để virus có thể khá chắc chắn rằng tệp đã bị nhiễm nếu các byte đó được đặt đúng cách được mã hóa. Lược đồ đơn giản nhất là chỉ cần đặt chúng ở một số vị trí cố định giá trị. Chúng ta sẽ sử dụng hai ký tự “VI” ở đây. Vì vậy, khi một tập tin bắt đầu bằng bước nhảy gần, theo sau là các byte “V”=56H và “I”=49H, chúng ta có thể gần như chắc chắn rằng có virus ở đó và nếu không thì không. Phải thừa nhận rằng, thỉnh thoảng virus sẽ khám phá tệp COM được thiết lập với bước nhảy theo sau là “VI” mặc dù nó không bị nhiễm bệnh. Khả năng điều này xảy ra Trường hợp thứ nhất: Kẻ lây nhiễm tệp COM đơn giản 37. Tuy nhiên, chúng rất nhỏ nên sẽ không có tổn thất lớn nếu virus không thể lây nhiễm một tập tin hiếm hoi này trong một triệu. Nó sẽ lây nhiễm sang mọi thứ khác.

Để đọc 5 byte đầu tiên của file, chúng ta mở nó bằng DOS Chức năng ngắt 21H 3D Hex. Chức năng này yêu cầu chúng ta thiết lập ds:dx để trỏ đến tên tệp (FNAME) và chỉ định quyền truy cập các quyền mà chúng tôi mong muốn trong sổ đăng ký al. Trong quy trình FILE\_OK virus chỉ cần đọc tập tin. Tuy nhiên, chúng tôi sẽ cố gắng mở nó với quyền truy cập đọc/ghi, thay vì quyền truy cập chỉ đọc. Nếu tập tin thuộc tính được đặt thành chỉ đọc, cố gắng mở ở chế độ đọc/ghi sẽ gây ra lỗi (điều mà DOS báo hiệu bằng cách đặt cờ nhớ khi trở về từ INT 21H). Điều này sẽ cho phép virus phát hiện các tệp chỉ đọc và tránh chúng, vì vi-rút phải ghi vào một tệp để lây nhiễm nó. Sẽ tốt hơn nhiều nếu biết rằng tập tin ở chế độ chỉ đọc ở đây, trong quy trình tìm kiếm, hơn là giả sử tệp có khả năng lây nhiễm tốt và sau đó virus sẽ thất bại khi nó thực sự cố gắng lây nhiễm. Như vậy khi mở file ta đặt al = 2 để báo cho DOS mở nó trong chế độ đọc/ghi. Nếu DOS mở file thành công, nó sẽ trả về một file xử lý trong rìu. Đây chỉ là một con số mà DOS sử dụng để chỉ tập tin trong tất cả các yêu cầu trong tương lai. Mã để mở tệp trông như thế này:

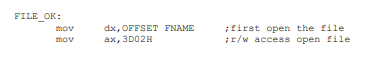


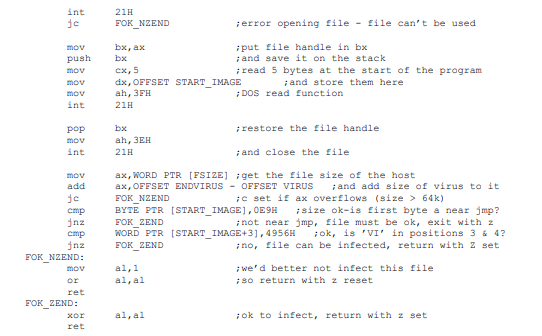
Khi tệp được mở, vi-rút có thể thực hiện việc đọc thực tế hoạt động, chức năng DOS 3F Hex. Để đọc một tập tin, người ta phải đặt bx bằng số xử lý tệp và cx bằng số byte cần đọc từ tập tin. Ngoài ra ds:dx phải được đặt ở vị trí trong bộ nhớ nơi lưu trữ dữ liệu đọc từ tệp (chúng tôi sẽ gọi START\_IMAGE). DOS lưu trữ một con trỏ tập tin nội bộ cho mỗi mở tệp để theo dõi vị trí trong tệp DOS sẽ thực hiện việc đọc và viết của nó từ đó. Con trỏ tệp chỉ dài bốn byte số nguyên, chỉ định byte nào trong tệp đã chọn được đọc hoặc hoạt động viết đề cập đến. Con trỏ tập tin này bắt đầu trỏ đến byte đầu tiên trong tệp (con trỏ tệp = 0) và nó tự động được được DOS nâng cao khi tệp được đọc hoặc ghi vào. Vì nó bắt đầu ở đầu tệp và thủ tục FILE\_OK phải đọc năm byte đầu tiên của tệp, không cần chạm vào tệp con trỏ ngay bây giờ. Tuy nhiên, bạn nên biết rằng nó ở đó, bị DOS ẩn đi. Nó là một phần thiết yếu của bất kỳ việc đọc và đọc tập tin nào. viết mà chúng tôi có thể muốn làm. Khi đến lúc virus xuất hiện lây nhiễm vào tập tin, nó sẽ phải sửa đổi con trỏ tập tin này để lấy một vài byte ở đây và đặt chúng ở đó, v.v. Làm như vậy nhanh hơn nhiều (vàdo đó, ít được chú ý hơn) so với việc đọc toàn bộ tệp vào bộ nhớ, thao tác nó trong bộ nhớ và sau đó ghi nó trở lại đĩa. Vì Tuy nhiên, hiện nay việc đọc tệp thực sự khá đơn giản. Nó trông như thế này:



Chúng tôi sẽ không lo lắng về khả năng xảy ra lỗi trong đọc năm byte ở đây. Lỗi duy nhất có thể xảy ra là tập tin không đủ dài để đọc năm byte và chúng tôi khá an toàn khi giả định rằng hầu hết các tệp COM sẽ có nhiều hơn bốn byte trong đó.

Cuối cùng, để đóng file, chúng ta sử dụng hàm DOS 3E Hex và đặt phần xử lý tập tin vào bx. Đặt tất cả lại với nhau, FILE\_OK thủ tục trông như thế này:



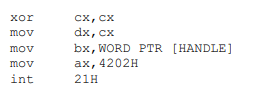


Điều này hoàn thành cuộc thảo luận của chúng ta về cơ chế tìm kiếm vi-rút.

**Cơ chế sao chép**

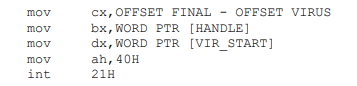
Sau khi virus tìm thấy một tập tin để lây nhiễm, nó phải thực hiện quá trình lây nhiễm. Chúng ta đã thảo luận ngắn gọn về cách thức thực hiện điều đó đã được hoàn thành, nhưng bây giờ hãy viết mã thực sự sẽ thực hiện Nó. Chúng ta sẽ đặt tất cả mã này vào một quy trình có tên là INFECT.

Mã cho INFECT khá đơn giản. Đầu tiên virus mở file có tên được lưu tại FNAME ở chế độ đọc/ghi chế độ, giống như khi tìm kiếm tệp và nó lưu trữ tệp xử lý trong vùng dữ liệu được gọi là HANDLE. Lần này, tuy nhiên chúng tôi muốn đi đến cuối tập tin và lưu trữ virus ở đó. Để làm như vậy, chúng tôi đầu tiên di chuyển con trỏ tập tin bằng hàm DOS 42H. Trong cuộc gọi chức năng 42H, thanh ghi bx phải được thiết lập với phần xử lý tệp số và cx:dx phải chứa số nguyên dài 32 bit cho biết vị trí để di chuyển con trỏ tập tin tới. Có ba cách khác nhau này chức năng có thể được sử dụng, như được chỉ định bởi nội dung của thanh ghi al. Nếu al=0, con trỏ tệp được đặt tương ứng với phần đầu của tệp. Nếu như al=1, nó được tăng lên tương ứng với vị trí hiện tại và nếu al=2, 40 Cuốn sách nhỏ về virus máy tính cx:dx được sử dụng làm phần bù từ cuối tệp. Kể từ lần đầu tiên Điều virus phải làm là đặt mã của nó vào cuối file COM nó đang tấn công, nó đặt con trỏ tập tin về cuối tập tin. Đây là dễ. Đặt cx:dx=0, al=2 và gọi hàm 42H:



Với con trỏ tập tin ở đúng vị trí, virus bây giờ có thể ghi chính nó ra đĩa ở cuối tập tin này. Để làm được điều đó, người ta chỉ đơn giản sử dụng chức năng ghi DOS, 40 Hex. Để sử dụng chức năng 40H người ta phải đặt ds:dx vào vị trí trong bộ nhớ nơi lưu trữ dữ liệu sẽ được ghi vào đĩa. Trong trường hợp này đó là sự khởi đầu của virus. Tiếp theo, đặt cx thành số byte cần ghi và bx cho phần xử lý tệp.

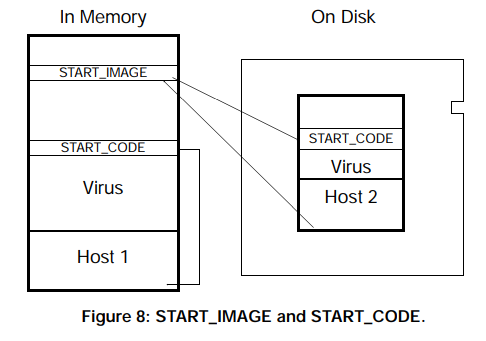
Có một vấn đề ở đây. Vì virus sẽ tự gắn vào các file COM với mọi kích cỡ khác nhau, địa chỉ của điểm bắt đầu của mã vi-rút không ở một vị trí cố định nào đó trong bộ nhớ. Mỗi tập tin được đính kèm sẽ đặt nó ở một nơi khác trong bộ nhớ. Vì vậy, virus phải đủ thông minh để tìm ra nó ở đâu. để làm điều này chúng ta sẽ sử dụng một thủ thuật trong quy trình điều khiển chính và lưu trữ phần bù của mã virus ở vị trí bộ nhớ có tên VIR\_START. Ở đây chúng tôi giả định rằng vị trí bộ nhớ này đã được khởi tạo đúng cách. Sau đó, đoạn mã ghi virus vào phần cuối của tập tin mà nó đang tấn công sẽ trông như thế này:

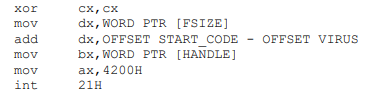


trong đó VIRUS là nhãn xác định điểm bắt đầu của mã virus và CUỐI CÙNG là nhãn xác định phần cuối của mã. Bù đắp CUỐI CÙNG - OFFSET VIRUS độc lập với vị trí của virus trong ký ức.

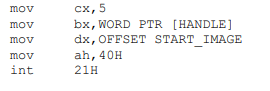
Bây giờ, với phần chính của mã virus được thêm vào cuối của tệp COM đang bị tấn công, virus phải thực hiện một số thao tác dọn dẹp công việc. Đầu tiên, nó phải di chuyển năm byte đầu tiên của tệp COM tới một vùng lưu trữ trong mã virus. Sau đó nó phải đặt lệnh nhảy cộng với các chữ cái mã 'VI' ở đầu tệp COM. Vì chúng tôi có đã đọc năm byte đầu tiên của tệp COM trong tìm kiếm thường lệ, họ đang sẵn sàng và chờ hành động lúc START\_IMAGE. Chúng ta chỉ cần ghi chúng ra đĩa ở vị trí thích hợp. Lưu ý rằng phải có hai vùng riêng biệt trong virus để lưu trữ năm vùng byte mã khởi động. Virus đang hoạt động phải có vùng dữ liệu START\_IMAGE để lưu trữ dữ liệu từ các tệp mà nó muốn lây nhiễm nhưng nó cũng phải có một khu vực khác mà chúng tôi sẽ gọi là START\_CODE. Cái này chứa năm byte đầu tiên của tệp mà nó thực sự được đính kèm. Nếu không có START\_CODE, vi-rút đang hoạt động sẽ không thể truyền kiểm soát chương trình chủ mà nó được gắn vào khi hoàn thành thực hiện.

Để ghi 5 byte đầu tiên của tập tin bị tấn công, virus phải lấy năm byte tại START\_IMAGE và lưu trữ chúng ở nơi START\_CODE nằm trên đĩa. Đầu tiên, virus thiết lập tập tin con trỏ tới vị trí của START\_CODE trên đĩa. Để tìm thấy điều đó vị trí, người ta phải lấy kích thước tệp gốc (được lưu tại FSIZE theo quy trình tìm kiếm) và thêm OFFSET START\_CODE - OFFSET VIRUS tới nó, di chuyển con trỏ tập tin về phía đầu của tập tin:

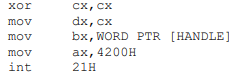




Tiếp theo, virus ghi 5 byte ở START\_IMAGE ra ngoài tài liệu:



Bước cuối cùng trong việc lây nhiễm một tập tin là thiết lập năm phần đầu tiên byte của tệp bằng cách nhảy tới đầu mã vi-rút, cùng với các chữ cái nhận dạng “VI”. Để làm được điều này, vị trí đầu tiên con trỏ tệp tới đầu tệp:



Tiếp theo, chúng ta phải thiết lập vùng dữ liệu trong bộ nhớ với đúng thông tin cần ghi vào đầu tập tin. START\_IMAGE là một nơi tốt để thiết lập các byte này vì dữ liệu ở đó không còn nữa cần thiết cho bất cứ điều gì. Byte đầu tiên phải là lệnh nhảy gần, E9 Hex:



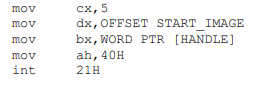
Hai byte tiếp theo phải là một từ để cho CPU biết cách thức nhiều byte để nhảy về phía trước. Byte này cần phải là tệp gốc kích thước của chương trình máy chủ, cộng với số byte trong virus trước khi bắt đầu mã thực thi (chúng tôi sẽ đặt một số dữ liệu Trường hợp thứ nhất: Kẻ lây nhiễm tệp COM đơn giản 43 ở đó). Chúng ta cũng phải trừ 3 từ số này vì bước nhảy tương đối luôn được tham chiếu đến con trỏ lệnh hiện tại, sẽ trỏ đến 103H khi bước nhảy thực sự được thực hiện. Do đó, hai byte cho chương trình biết vị trí nhảy được thiết lập qua



Cuối cùng thiết lập các byte ID 'VI' trong vùng dữ liệu năm byte của chúng tôi,



ghi dữ liệu vào đầu tập tin, sử dụng chức năng ghi của DOS,



rồi đóng tệp bằng DOS,

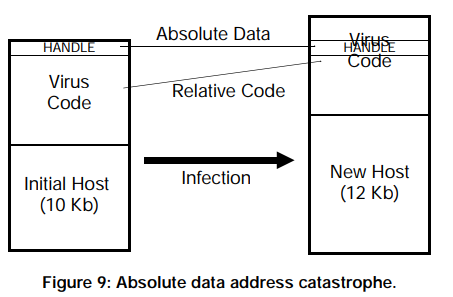


Điều này hoàn thành cơ chế sao chép.

**Lưu trữ dữ liệu cho virus**

Một vấn đề chúng ta phải đối mặt khi tạo ra loại virus này là làm thế nào để định vị dữ liệu. Vì tất cả các bước nhảy và lệnh gọi trong tệp COM đều mang tính tương đối, nên chúng ta không cần phải làm bất cứ điều gì cầu kỳ để giải thích thực tế là virus phải tự di chuyển khi nó tự sao chép từ chương trình này sang chương trình khác. các nhảy và các cuộc gọi tự động di chuyển. Xử lý dữ liệu không hề dễ dàng. Một tham chiếu dữ liệu như





đề cập đến độ lệch tuyệt đối trong phân đoạn chương trình có nhãn HANDLE. Chúng ta không thể chỉ định nghĩa một từ trong bộ nhớ bằng trình biên dịch mã chỉ thị như



rồi tập hợp virus và chạy nó. Nếu chúng ta làm điều đó thì nó sẽ thành công đúng ngay lần đầu tiên. Một khi nó đã tự gắn vào một chương trình mới, Tuy nhiên, tất cả địa chỉ bộ nhớ sẽ thay đổi và virus sẽ gặp rắc rối lớn. Nó sẽ tự đánh bom hoặc khiến vật chủ của nó chương trình đánh bom.

Có hai cách để tránh thảm họa ở đây. Đầu tiên, một có thể đặt tất cả dữ liệu lại với nhau ở một nơi và viết chương trình để tự động xác định vị trí của dữ liệu và lưu trữ giá trị đó trong một thanh ghi (ví dụ: si) để truy cập nó một cách linh hoạt, như thế này:



trong đó HANDLE\_OFS là phần bù của biến HANDLE từ điểm bắt đầu của vùng dữ liệu.

Ngoài ra, chúng ta có thể đặt tất cả dữ liệu vào một vị trí cố định trong đoạn mã, miễn là chúng tôi chắc chắn rằng cả virus và máy chủ sẽ chiếm không gian đó. Nơi an toàn duy nhất để làm điều này nằm ở cuối đoạn, nơi chứa ngăn xếp. Vì virus chiếm quyền kiểm soát CPU trước tiên khi tệp COM được thực thi, nó cũng sẽ kiểm soát ngăn xếp. Vì vậy chúng ta có thể xác định chính xác những gì ngăn xếp đang hoạt động và hãy tránh xa nó. Đây là phương pháp chúng tôi chọn.

Khi virus giành được quyền kiểm soát lần đầu tiên, con trỏ ngăn xếp sp sẽ được được đặt thành FFFF Hex. Nếu nó gọi một chương trình con, địa chỉ ngay sau cuộc gọi được đặt trên ngăn xếp, theo byte FFFF Hex và FFFE Hex trong phân đoạn của chương trình và con trỏ ngăn xếp bị giảm đi bằng hai, thành FFFD Hex. Khi CPU thực thi lệnh quay về trong chương trình con, nó sử dụng hai byte được lưu trữ bởi lệnh gọi tới xác định nơi quay lại và tăng con trỏ ngăn xếp lên hai. Tương tự như vậy, việc thực hiện lệnh đẩy sẽ làm giảm ngăn xếp bằng hai byte và lưu trữ thanh ghi mong muốn tại vị trí của con trỏ ngăn xếp. Lệnh pop đảo ngược quá trình này. int lệnh yêu cầu năm byte không gian ngăn xếp và điều này bao gồm các lệnh gọi tới các trình xử lý ngắt phần cứng, có thể được truy cập bất cứ lúc nào trong chương trình mà không có cảnh báo, cái này chồng lên cái kia.

Vùng dữ liệu của virus có thể nằm ngay bên dưới bộ nhớ cần thiết cho ngăn xếp. Lượng không gian ngăn xếp chính xác yêu cầu khá khó xác định, nhưng 80 byte sẽ nhiều hơn quá đủ. Dữ liệu sẽ ở ngay bên dưới 80 byte này và trong bằng cách này, vị trí của nó có thể được cố định. Người ta chỉ cần tính đến không gian nó chiếm khi xác định kích thước tối đa của một COM trong quy trình FILE\_OK.

Tất nhiên, người ta không thể đặt các biến khởi tạo vào ngăn xếp. Chúng phải được lưu trữ cùng với chương trình trên đĩa. Để lưu trữ chúng gần đoạn cuối của chương trình sẽ yêu cầu virus mở rộng kích thước tệp của mỗi tệp gần đến giới hạn 64K. Sự thay đổi mạnh mẽ như vậy ở kích thước tệp sẽ nhanh chóng báo cho người dùng biết rằng hệ thống của anh ta đã bị bị lây nhiễm! Thay vào đó, các biến khởi tạo nên được lưu trữ với mã virus thực thi được. Chiến lược này sẽ giữ số byte phải được thêm vào máy chủ ở mức tối thiểu. (Vì vậy nó là một biện pháp chống phát hiện đáng giá.) Hạn chế là như vậy các biến sau đó phải được virus định vị một cách linh hoạt trong thời gian chạy.

May mắn thay, chúng ta chỉ có một phần dữ liệu cần được được khởi tạo trước, chuỗi được DOS sử dụng trong thủ tục tìm kiếm để định vị các tệp COM mà chúng tôi gọi đơn giản là “COMFILE”. Nếu bạn lấy nhìn lại quy trình tìm kiếm, bạn sẽ nhận thấy rằng chúng tôi đã thực hiện 46 Cuốn sách nhỏ về virus máy tính khả năng tái định vị của phần dữ liệu này được tính đến khi chúng tôi lấy nó bằng cách sử dụng các hướng dẫn



thay vì đơn giản



**Quy trình kiểm soát chính**

Bây giờ chúng ta đã có đầy đủ công cụ để viết virus TIMID. Tất cả điều cần thiết là một thói quen kiểm soát tổng thể để kéo mọi thứ cùng nhau. Thói quen chính này phải:

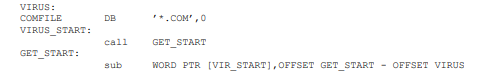
1) Tự động xác định vị trí (độ lệch) của virus trong bộ nhớ.

2) Gọi quy trình tìm kiếm để tìm một chương trình mới để lây nhiễm.

3) Lây nhiễm vào chương trình được định vị bởi quy trình tìm kiếm, nếu nó đã tìm thấy một cái.

4) Trả lại quyền điều khiển cho chương trình chủ.

Để xác định vị trí của virus trong bộ nhớ, chúng tôi sử dụng một thủ thuật đơn giản. Lệnh đầu tiên trong quy trình điều khiển chính sẽ trông như thế này:



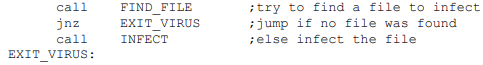
Cuộc gọi đẩy địa chỉ tuyệt đối của GET\_START vào ngăn xếp tại FFFC Hex (vì đây là hướng dẫn đầu tiên của virus và lệnh đầu tiên sử dụng ngăn xếp). Tại vị trí đó, chúng tôi phủ lên ngăn xếp với một biến từ có tên là VIR\_START. Sau đó chúng tôi trừ sự khác biệt về độ lệch giữa GET\_START và byte đầu tiên của loại virus có tên VIRUS. Thủ thuật lập trình đơn giản này có được Trường hợp thứ nhất: Kẻ lây nhiễm tệp COM đơn giản 47 độ lệch tuyệt đối của byte đầu tiên của virus trong phân đoạn chương trình, và lưu trữ nó trong một biến dễ dàng truy cập.

Tiếp theo là bước chống phát hiện quan trọng: Trình chủ quy trình điều khiển di chuyển Vùng truyền đĩa (DTA) sang vùng dữ liệu đối với virus sử dụng chức năng DOS 1A Hex,



Động thái này là cần thiết vì quá trình tìm kiếm sẽ sửa đổi dữ liệu trong DTA. Khi tệp COM khởi động, DTA được đặt thành mặc định giá trị của độ lệch 80 H trong phân đoạn chương trình. Vấn đề là rằng nếu chương trình chủ yêu cầu các tham số dòng lệnh, chúng được lưu trữ cho chương trình tại cùng vị trí này. Nếu DTA là không bị thay đổi tạm thời trong khi virus đang thực thi, việc tìm kiếm thường trình sẽ ghi đè lên bất kỳ tham số dòng lệnh nào trước chương trình chủ đã có cơ hội truy cập chúng. Điều đó sẽ gây ra bất kỳ chương trình COM bị nhiễm yêu cầu tham số dòng lệnh để đánh bom. Virus sẽ hoạt động bình thường và lưu trữ các chương trình không yêu cầu tham số nào sẽ chạy tốt, nhưng người dùng có thể phát hiện rắc rối với một số chương trình. Tạm thời di chuyển DTA sẽ loại bỏ vấn đề này.

Khi DTA được di chuyển, quy trình điều khiển chính có thể được thực hiện một cách an toàn gọi các thủ tục tìm kiếm và sao chép:



Cuối cùng, thủ tục điều khiển chính phải trả lại quyền điều khiển cho máy chủ chương trình. Điều này bao gồm ba bước: Thứ nhất, khôi phục DTA về trạng thái ban đầu của nó. giá trị ban đầu, bù 80H,

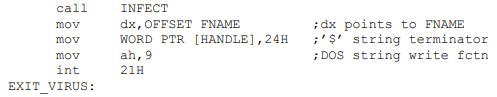


Tiếp theo, di chuyển năm byte đầu tiên của chương trình máy chủ ban đầu từ vùng dữ liệu START\_CODE nơi chúng được lưu trữ ở đầu chương trình chủ ở 100H, Cuối cùng virus phải chuyển quyền điều khiển sang chương trình chủ ở 100H. Điều này đòi hỏi một thủ thuật, vì người ta không thể đơn giản nói “jmp 100H” vì bước nhảy như vậy là tương đối nên lệnh đó sẽ không nhảy tới 100H ngay khi virus di chuyển sang tệp khác và điều đó báo hiệu thảm họa. Một lệnh chuyển điều khiển tới một độ lệch tuyệt đối là kết quả trả về từ một cuộc gọi. Vì chúng tôi đã thực hiện cuộc gọi ngay lúc sự bắt đầu của quy trình điều khiển chính và chúng tôi chưa thực hiện chưa trả về tương ứng, việc thực hiện lệnh re sẽ cả hai chuyển quyền điều khiển đến máy chủ và nó sẽ xóa ngăn xếp. Tất nhiên rồi, địa chỉ trả về phải được đặt thành 100H để chuyển quyền điều khiển tới máy chủ chứ không phải ở nơi nào khác. Địa chỉ trả lại đó chỉ là từ tại VIR\_START. Vì vậy, để chuyển quyền điều khiển cho máy chủ, chúng ta viết



Bingo, chương trình chủ tiếp quản và chạy như thể virus chưa từng có đã ở đó.

Như đã viết, quy trình kiểm soát tổng thể này hơi nguy hiểm, bởi vì nó sẽ làm cho virus hoàn toàn vô hình đối với người dùng khi anh ta chạy một chương trình... để nó có thể thoát khỏi. Có vẻ khôn ngoan để chế ngự quái thú một chút khi chúng ta mới bắt đầu. Vì vậy, sau lệnh gọi INFECT,chúng ta hãy thêm một vài dòng vào để hiển thị tên của tệp virus vừa bị nhiễm:



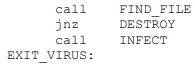
Cái này sử dụng hàm DOS 9 để in chuỗi tại FNAME, đó là tên file bị nhiễm. Lưu ý rằng nếu ai đó muốn để tạo ra một con quái vật độc hại từ virus này, mã hủy diệt có thể dễ dàng được đặt ở đây hoặc sau EXIT\_VIRUS, tùy thuộc vào điều kiện mà hoạt động phá hoại được mong muốn. Ví dụ, hacker của chúng ta có thể viết một thủ tục gọi là DESTROY, sẽ gây ra mọi sự tàn phá, và sau đó mã hóa nó như thế này:



nếu một người chỉ muốn gây sát thương sau khi lây nhiễm thành công địa điểm, hoặc như thế này:



nếu một người muốn thiệt hại luôn xảy ra, bất kể thế nào, hoặc như thế này:



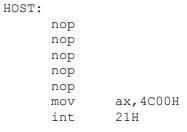
nếu người ta muốn thiệt hại chỉ xảy ra trong trường hợp virus có thể không tìm thấy tập tin để lây nhiễm, v.v., v.v. Tôi nói điều này không có nghĩa là bạn viết một thói quen như vậy—xin đừng—nhưng chỉ để cho bạn thấy nó dễ dàng như thế nào nó sẽ là để kiểm soát hành vi phá hoại của virus (hoặc bất kỳ loại nào khác chương trình, cho vấn đề đó).

**Máy chủ đầu tiên**

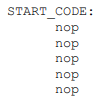
Để biên dịch và chạy virus, virus phải được gắn vào máy chủ chương trình. Nó không thể tồn tại một mình. Bằng văn bản hợp ngữ mã cho virus này, chúng ta phải thiết lập mọi thứ để virus nghĩ nó đã được đính kèm vào một số tập tin COM. Tất cả những gì cần thiết là một cách đơn giản chương trình không làm gì khác ngoài việc thoát sang DOS. Để trả lại quyền điều khiển cho DOS, một chương trình thực thi chức năng DOS 4C Hex. Điều đó chỉ dừng lại chương trình ngừng chạy và DOS tiếp quản. Khi chức năng 4C được thực thi, một mã trả về sẽ được đưa vào bởi chương trình thực hiện cuộc gọi, trong đó al=0 cho biết chương trình đã hoàn thành thành công. Bất kỳ giá trị nào khác đều chỉ ra một loại lỗi nào đó, do chương trình thực hiện lệnh gọi DOS xác định. Vì vậy, chương trình COM đơn giản nhất sẽ trông như thế này:



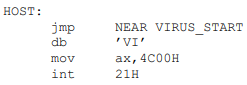
Vì virus sẽ chiếm lấy 5 byte đầu tiên của COM và vì có thể bạn không biết tập tin trên có bao nhiêu byte hai hướng dẫn sẽ chiếm, hãy đặt năm NOP (không hoạt động) hướng dẫn khi bắt đầu chương trình máy chủ. Chúng chiếm năm byte không làm gì cả. Như vậy chương trình chủ sẽ có dạng như sau:



Tuy nhiên, chúng tôi không muốn mã hóa nó như vậy! Chúng tôi mã hóa nó thành trông giống như bị virus lây nhiễm vậy. Cụ thể là NOP sẽ được lưu trữ tại START CODE,



và năm byte đầu tiên của máy chủ sẽ bao gồm bước nhảy tới virus và các chữ cái “VI”:



Đó, thế thôi. Virus TIMID được liệt kê đầy đủ trong Phụ lục A, cùng với mọi thứ bạn cần để biên dịch nó một cách chính xác.

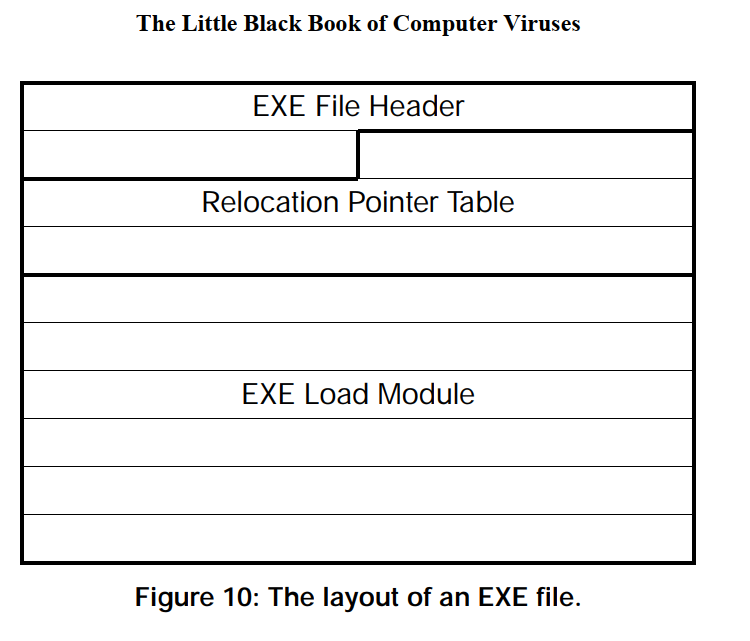
Tôi nhận ra rằng bạn có thể bị choáng ngợp bởi những ý tưởng mới và các chi tiết kỹ thuật ở thời điểm này, và theo tôi gọi loại virus này là “đơn giản” có thể làm nản lòng. Nếu vậy, đừng mất lòng. Nghiên cứu nó cẩn thận. Quay lại văn bản và ghép các phần khác nhau lại với nhau các yếu tố chức năng, từng cái một. Và nếu bạn cảm thấy tự tin, bạn có thể thử đặt nó vào thư mục con của chính nó trên máy của bạn và làm cho nó quay cuồng. Nếu bạn làm vậy, hãy cẩn thận! Tiến hành tại của bạn rủi ro của chính mình! Nó không giống bất kỳ chương trình máy tính nào khác mà bạn từng chạy!

**Một loại virus thực thi tinh vi**

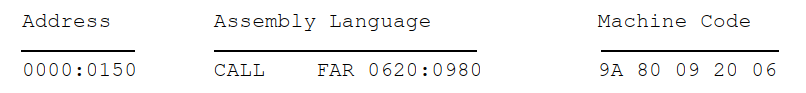
Phần mềm lây nhiễm tập tin COM đơn giản mà chúng tôi vừa phát triển có thể là hướng dẫn tốt về những điều cơ bản về cách viết virus, nhưng nó bị hạn chế nghiêm trọng. Vì nó chỉ tấn công các tập tin COM trong hiện tại thư mục, nó sẽ gặp khó khăn trong việc phát triển. Trong chương này, chúng tôi sẽ phát triển một loại virus phức tạp hơn có thể vượt qua được những hạn chế. . . . một loại virus có thể lây nhiễm các tệp EXE và thư mục nhảy vào thư mục và lái xe vào ổ đĩa. Những cải tiến như vậy làm cho virus phức tạp hơn nhiều và cũng nguy hiểm hơn nhiều. chúng tôi bắt đầu với một điều gì đó đơn giản và tương đối vô thưởng vô phạt ở chương trước. Bạn không thể gặp quá nhiều rắc rối với nó. Tuy nhiên, tôi không muốn để lại cho bạn đồ chơi trẻ em duy nhất. Loại virus mà chúng ta thảo luận trong phần này chương có tên là INTRUDER, không phải đồ chơi. Nó rất có khả năng tìm thấy nó xâm nhập vào các máy tính trên khắp thế giới và đánh lừa rất nhiều người máy tính có khả năng whiz.

**Cấu trúc của một tệp tin EXE**

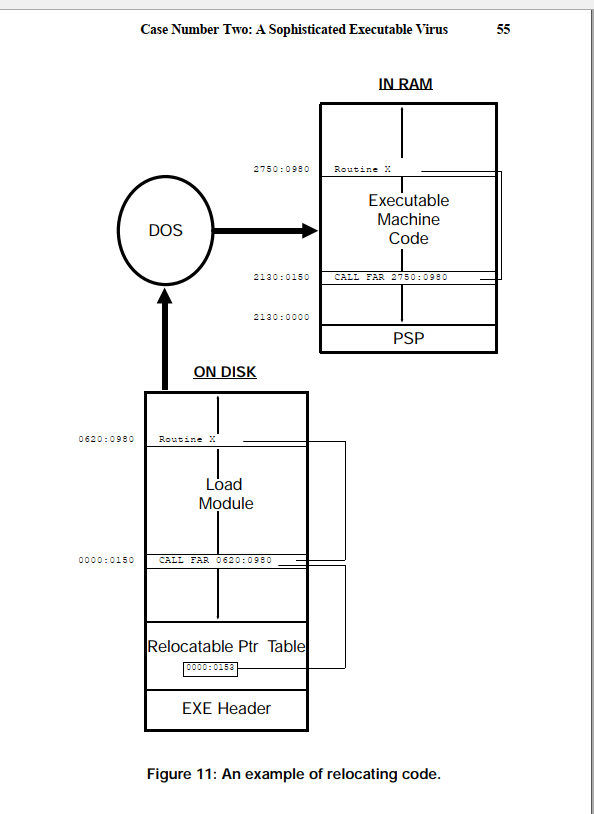
Tệp EXE không đơn giản như tệp COM. Tệp EXE được thiết kế để cho phép DOS thực thi các chương trình đòi hỏi nhiều hơn 64 kilobyte mã, dữ liệu và ngăn xếp. Khi tải EXE tệp, DOS không đưa ra giả định tiên nghiệm nào về kích thước của tệp,hoặc mã hoặc dữ liệu là gì. Tất cả thông tin này được lưu trữ trong EXE chính tệp đó, trong Tiêu đề EXE ở đầu tệp.

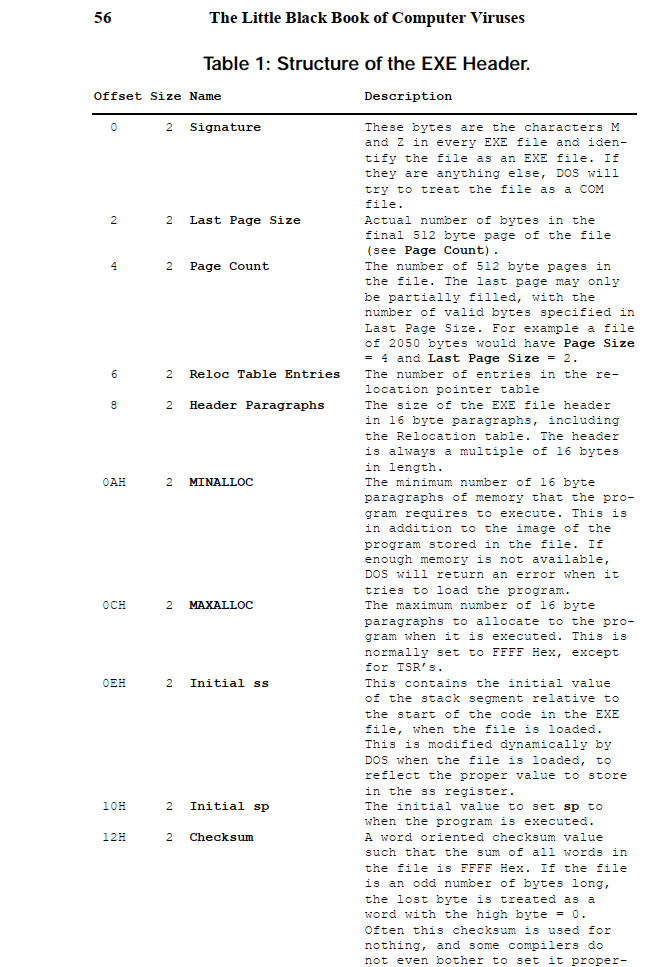


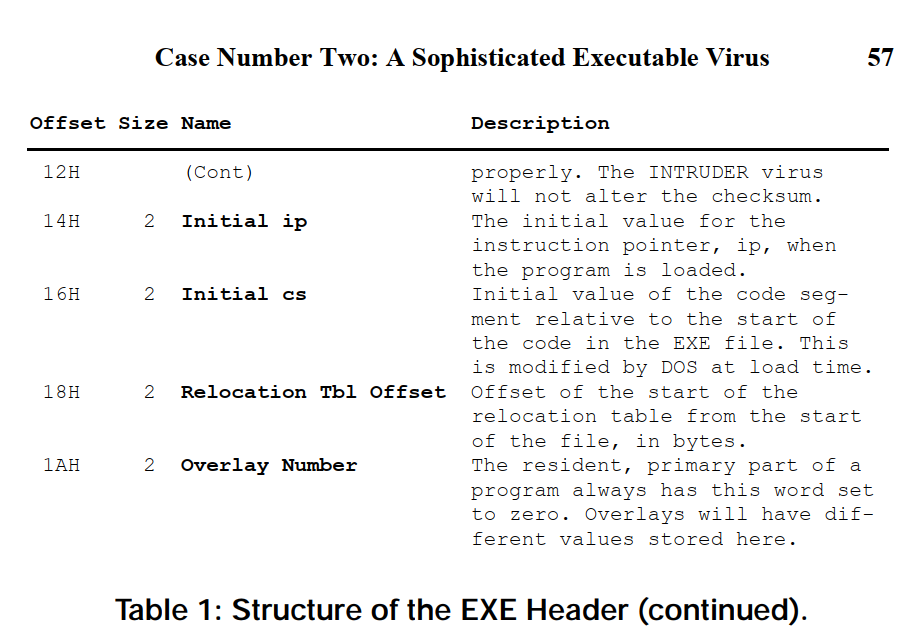
Tiêu đề này có hai phần, một phần có độ dài cố định và một biến bảng độ dài của các con trỏ tới các tham chiếu phân đoạn trong Mô-đun tải, được gọi là Bảng con trỏ tái định vị. Vì bất kỳ virus nào tấn công. Các tệp EXE phải có khả năng thao tác dữ liệu trong Tiêu đề EXE, tốt hơn chúng ta nên dành chút thời gian để xem xét nó. Hình 10 là đồ họađại diện của một tập tin EXE. Ý nghĩa của từng byte trong tiêu đề được giải thích trong Bảng 1. Khi DOS tải EXE, nó sử dụng Con trỏ định vị lại Bảng để sửa đổi tất cả các tham chiếu phân đoạn trong Mô-đun tải. Sau đó rằng, các tham chiếu phân đoạn trong hình ảnh của chương trình được tải vào điểm nhớ tới vị trí bộ nhớ chính xác. Hãy xem xét một ví dụ (Hình 11): Hãy tưởng tượng một tệp EXE có hai phân đoạn. Phân đoạn ở đầu mô-đun tải chứa lệnh gọi xa tới đoạn thứ hai. Trong mô-đun tải, cuộc gọi này trông như thế này:



Từ đó, người ta có thể suy ra rằng điểm bắt đầu của đoạn thứ hai là 6200H (= 620H x 10H) byte kể từ khi bắt đầu mô-đun tải.







Bảng con trỏ tái định vị sẽ chứa một vectơ 0000:0153 tới điểm tới tham chiếu phân đoạn (20 06) của lệnh gọi xa này. Khi DOS tải chương trình, nó có thể tải nó bắt đầu từ phân đoạn 2130H, bởi vì DOS và một số chương trình lưu trú trong bộ nhớ chiếm các vị trí bên dưới cái này. Vì vậy, trước tiên DOS sẽ tải Mô-đun tải vào bộ nhớ tại 2130:0000. Sau đó nó sẽ lấy con trỏ tái định vị 0000:0153 và biến nó thành một con trỏ, 2130:0153 trỏ đến phân đoạn trong lệnh gọi xa trong bộ nhớ. DOS sau đó sẽ thêm 2130H vào từ ở vị trí đó, dẫn đến mã ngôn ngữ máy là 9A 80 09 50 27, hoặc GỌI FAR 2750:0980 (Xem Hình 11). Lưu ý rằng chương trình COM không yêu cầu các bài tập thể dục trị liệu này vì nó không chứa tham chiếu phân đoạn. Vì vậy, DOS chỉ cần đặt tất cả các thanh ghi phân đoạn thành một giá trị trước khi chuyển điều khiển tới chương trình.

**Lây nhiễm vào một tệp tin EXE**

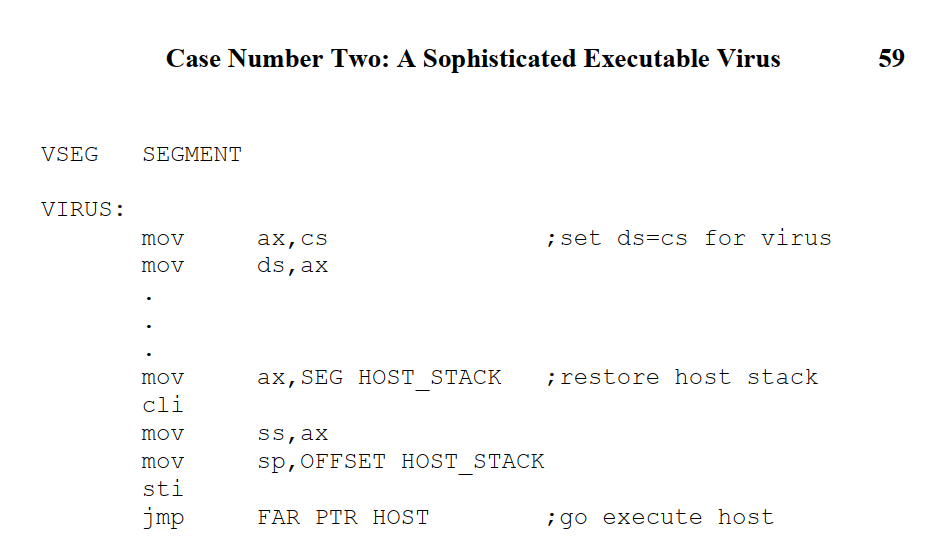
Một loại vi-rút muốn lây nhiễm vào tệp EXE sẽ phải cũng sửa đổi Tiêu đề EXE và Bảng con trỏ định vị lại như thêm mã riêng của nó vào Mô-đun tải. Điều này có thể được thực hiện trong một theo nhiều cách khác nhau, một số cách đòi hỏi nhiều công sức hơn người khác. Virus INTRUDER sẽ tự gắn vào phần cuối của EXE chương trình và giành quyền kiểm soát khi chương trình bắt đầu lần đầu tiên. Điều này sẽ trường hợp thứ hai: Một loại virus thực thi tinh vi 57 yêu cầu một quy trình tương tự như trong TIMID, sao chép chương trình mã từ bộ nhớ vào một tệp trên đĩa, sau đó điều chỉnh tệp.

INTRUDER sẽ có mã, dữ liệu và ngăn xếp riêng phân đoạn. Một virus EXE phổ quát không thể đưa ra bất kỳ giả định nào cách các phân đoạn đó được thiết lập bởi chương trình chủ. Nó sẽ sụp đổ ngay khi nó tìm thấy một chương trình có những giả định đó bị vi phạm. Ví dụ: nếu người ta sử dụng bất kỳ ngăn xếp nào của máy chủ chương trình được khởi tạo, ngăn xếp có thể kết thúc ngay trong giữa mã virus với máy chủ phù hợp. (Ký ức đó sẽ đã có không gian trống trước khi virus lây nhiễm vào chương trình.) Như ngay khi virus bắt đầu thực hiện cuộc gọi hoặc đẩy dữ liệu lên ngăn xếp, nó sẽ làm hỏng mã của chính nó và tự hủy.

Để thiết lập các phân đoạn cho virus, các giá trị phân đoạn ban đầu mới đối với cs và ss phải được đặt trong tiêu đề tệp EXE. Ngoài ra, cái cũ các phân đoạn ban đầu phải được lưu trữ ở đâu đó trong virus, vì vậy nó có thể chuyển quyền điều khiển trở lại chương trình chủ khi nó thực thi xong. Chúng ta sẽ phải đặt hai con trỏ tới các tham chiếu phân đoạn này trong bảng con trỏ tái định vị, vì chúng là các tham chiếu có thể định vị lại bên trong đoạn mã virus.

Việc thêm con trỏ vào bảng con trỏ tái định vị sẽ xuất hiện một câu hỏi quan trọng Để thêm con trỏ vào con trỏ tái định vị bảng, đôi khi có thể cần phải mở rộng kích thước của bảng đó. Vì Tiêu đề EXE phải có kích thước bội số của 16 byte, con trỏ tái định vị được phân bổ theo khối bốn con trỏ bốn byte. Vì vậy, nếu chúng ta có thể giữ số lượng tham chiếu phân đoạn xuống hai, sẽ chỉ cần mở rộng tiêu đề mỗi lần. Mặt khác, virus có thể chọn không lây nhiễm vào tệp, thay vào đó hơn là mở rộng tiêu đề. Có những ưu và nhược điểm cho cả hai khả năng. Một mặt, một mô-đun tải có thể có hàng trăm hàng kilobyte và việc di chuyển nó là một công việc tốn thời gian có thể thể hiện rất rõ ràng rằng có điều gì đó không nên xảy ra. Mặt khác, nếu virus chọn không di chuyển mô-đun tải, thì khoảng một nửa số tệp EXE sẽ miễn nhiễm một cách tự nhiên với sự nhiễm trùng. Virus INTRDER sẽ im lặng và thận trọng cách tiếp cận không lây nhiễm sang mọi EXE. Bạn có thể muốn thử cách tiếp cận khác như một bài tập và chỉ di chuyển mô-đun tải khi cần thiết và chỉ dành cho các tệp tương đối nhỏ (chọn kích thước tối đa).

Giả sử quy trình virus chính trông giống như thế này:



Sau đó, để lây nhiễm một tập tin mới, quá trình sao chép phải thực hiện các bước sau:

1. Đọc Tiêu đề EXE trong chương trình chủ.

2. Mở rộng kích thước của mô-đun tải cho đến khi nó bằng phẳng bội số của 16 byte, vì vậy cs:0000 sẽ là byte đầu tiên của vi-rút.

3. Viết mã virus hiện đang thực thi vào cuối tập tin EXE đang bị tấn công.

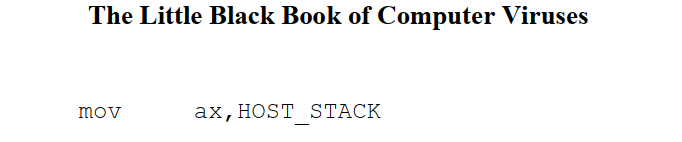
4. Viết các giá trị ban đầu của ss:sp, như được lưu trong EXE Tiêu đề, tới vị trí của SEG HOST\_STACK và OFFSET HOST\_STACK trên đĩa theo đoạn mã trên.

5. Viết giá trị ban đầu của cs:ip trong Tiêu đề EXE vào vị trí của FAR PTR HOST trên đĩa ở trên mã số.

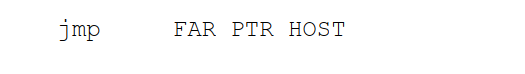
6. Lưu trữ ban đầu ss=SEG VSTACK, ban đầu sp=OFFSET VSTACK, viết tắt cs=SEG VSEG, và viết tắt ip=OFFSET VIRUS trong tiêu đề EXE thay cho những giá trị cũ.

7. Thêm hai mục vào Bảng định vị lại trong EXE tiêu đề.

8. Thêm hai con trỏ tái định vị vào cuối Bảng con trỏ tái định vị trong tệp EXE trên đĩa (vị trí của các con trỏ này được tính từ tiêu đề). Các con trỏ đầu tiên phải trỏ đến SEG HOST\_STACK trong chỉ dẫn



Phần thứ hai sẽ trỏ đến phần phân đoạn của hướng dẫn trong quy trình virus chính.



9. Tính toán lại kích thước của tệp EXE bị nhiễm và điều chỉnh các trường tiêu đề Số trang và Trang cuối Kích thước phù hợp.

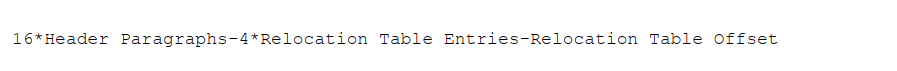
10. Ghi lại Tiêu đề EXE mới vào đĩa.

Tất cả các giá trị phân đoạn ban đầu phải được tính từ kích thước của mô-đun tải đang bị lây nhiễm. Mã để hoàn thành nhiễm trùng này nằm trong INFECT thông thường trong Phụ lục B.

**Một cơ chế tìm kiếm tệp tin liên tục**

Giống như virus TIMID, cơ chế tìm kiếm có thể được chia thành hai phần: FIND\_FILE chỉ xác định các vị trí có thể tập tin để lây nhiễm. FILE\_OK, xác định liệu một tập tin có thể bị nhiễm virus hay không.

Thủ tục FILE\_OK sẽ gần giống như thủ tục một trong TIMID. Nó phải mở tập tin được đề cập và xác định liệu nó có thể bị nhiễm hay không và đảm bảo rằng nó chưa bị lây nhiễm. Hai tiêu chí duy nhất để xác định liệu một tệp EXE có thể bị nhiễm là Số Lớp phủ có bằng 0 hay không và liệu nó có đủ chỗ trong bảng con trỏ tái định vị cho hai nhiều con trỏ hơn. Yêu cầu thứ hai được xác định bởi một đơn giản tính toán từ các giá trị được lưu trữ trong tiêu đề EXE. Nếu như



lớn hơn hoặc bằng 8 (=4 lần số lượng thiết bị định vị lại virus yêu cầu), thì có đủ chỗ trong việc di dời bảng con trỏ. Việc tính toán này được thực hiện bởi chương trình con REL\_ROOM, được gọi bởi FILE\_OK.

Để xác định xem virus đã lây nhiễm vào một tập tin hay chưa, chúng tôi đặt một từ ID có giá trị được gán trước vào đoạn mã 60 Cuốn sách nhỏ về virus máy tính màu đen ở một mức chênh lệch cố định (giả sử là 0). Sau đó, khi kiểm tra tệp, FILE\_OK lấy phân đoạn từ cs ban đầu trong tiêu đề EXE. Nó dùng cái đó với độ lệch 0 để tìm từ ID trong mô-đun tải (được cung cấp có virus ở đó). Nếu virus chưa lây nhiễm vào tập tin, cs ban đầu sẽ chứa đoạn mã ban đầu của chương trình máy chủ. Sau đó phép tính của chúng tôi sẽ lấy một số từ ngẫu nhiên ra khỏi tệp có thể sẽ không khớp với giá trị được yêu cầu của từ ID. Trong này cách FILE\_OK sẽ biết rằng tập tin đó không bị nhiễm virus. Vì thế FILE\_OK vẫn khá đơn giản.

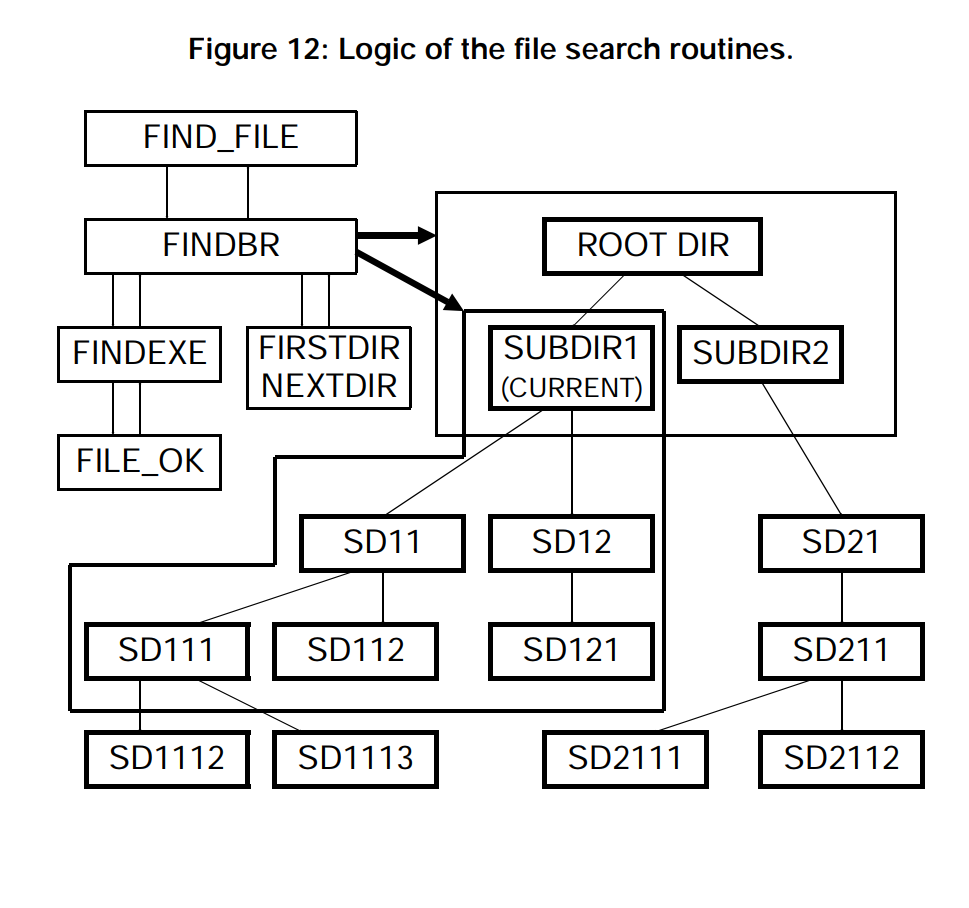
Tuy nhiên, chúng tôi muốn thiết kế phức tạp hơn nhiều Thủ tục FIND\_FILE hơn TIMID. Quy trình trong TIMID chỉ có thể tìm kiếm các tập tin trong thư mục hiện tại để tấn công. Cái đó lúc đầu thì ổn, nhưng một loại virus tốt sẽ có thể nhảy từ thư mục này sang thư mục khác và thậm chí từ ổ này sang ổ khác. Chỉ trong này Làm cách nào để virus có cơ hội hợp lý để lây nhiễm một lượng lớn một phần tệp trên hệ thống và chuyển từ hệ thống này sang hệ thống khác.

Để tìm kiếm nhiều thư mục, chúng ta cần tìm kiếm theo cây lịch trình. Đó là một thuật toán khá phổ biến trong lập trình. Chúng tôi viết một thường trình FIND\_BR, với một thư mục sẽ tìm kiếm nó đối với EXE sẽ vượt qua FILE\_OK. Nếu nó không tìm thấy một tập tin, nó sẽ tiến hành tìm kiếm các thư mục con của thư mục hiện được tham chiếu thư mục. Đối với mỗi thư mục con được tìm thấy, FIND\_BR sẽ đệ quy gọi chính nó bằng cách sử dụng thư mục con mới làm thư mục để thực hiện tìm kiếm trên. Theo cách này, tất cả các thư mục con của bất kỳ thư mục nào thư mục có thể được tìm kiếm để tìm kiếm một tập tin lây nhiễm. Nếu người ta chỉ định thư mục để tìm kiếm như thư mục gốc thì tất cả các tập tin trên đĩa sẽ được tìm kiếm.

Thực hiện tìm kiếm quá dài và liên quan có thể là một vấn đề mặc dù. Một đĩa cứng lớn có thể dễ dàng chứa hàng trăm thư mục con và hàng nghìn tệp tin. Khi virus mới xâm nhập vào hệ thống, nó sẽ nhanh chóng tìm thấy một tập tin không bị nhiễm virus mà nó có thể tấn công, vì vậy việc tìm kiếm sẽ nhanh đến mức không thể nhận thấy. Tuy nhiên, một khi hầu hết các tập tin trên hệ thống đã bị nhiễm virus, virus có thể làm cho đĩa kêu vo vo trong hai mươi giây trong khi kiểm tra tất cả EXE trên một trang nhất định lái xe đi tìm một người để lây nhiễm. Đó có thể là một đầu mối khá rõ ràng rằng có gì đó không ổn.

Để giảm thiểu thời gian tìm kiếm, chúng ta phải cắt ngắn tìm kiếm theo cách mà virus vẫn có cơ hội hợp lý để lây nhiễm Trường hợp thứ hai: Một loại virus thực thi tinh vi 61 lây nhiễm vào mọi tệp EXE trên hệ thống. Để làm điều đó chúng tôi sử dụng thói quen của người dùng PC điển hình. Thông thường, EXE được lan truyền khá rộng rãi. đồng đều trên khắp các thư mục khác nhau. Người dùng thường xuyên đặt các chương trình đã sử dụng trong đường dẫn của chúng và thực thi chúng từ các thư mục. Vì vậy, nếu vi-rút của chúng tôi tìm kiếm thư mục hiện tại và tất cả trong số các thư mục con của nó, sâu tới hai cấp độ, nó sẽ có vị trí tốt có khả năng lây nhiễm toàn bộ đĩa. Khi được bổ sung thêm bảo hiểm, nó cũng có thể tìm kiếm thư mục gốc và tất cả các thư mục con của nó ở một cấp độ sâu. Rõ ràng, virus sẽ có thể di chuyển sang các ổ đĩa khác nhau và các thư mục mà không cần tìm kiếm chúng một cách cụ thể, bởi vì nó sẽ tấn công các tập tin trên ổ đĩa hiện tại khi một chương trình bị nhiễm được thực thi và chương trình được thực thi không cần phải ở trạng thái hiện tại lái xe.

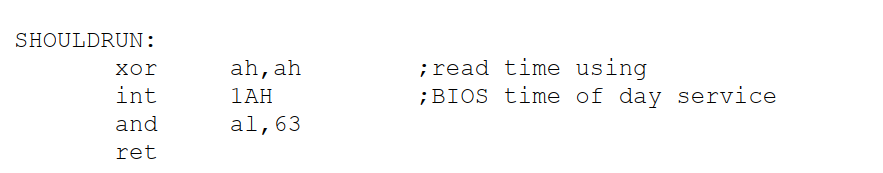
Khi mã hóa thủ tục FIND\_FILE, sẽ thuận tiện hơn khi cấu trúc nó theo ba cấp độ. Đầu tiên là thủ tục chính FIND\_FILE, quyết định nhánh thư mục con nào cần tìm kiếm. thứ hai cấp độ là một thói quen sẽ tìm kiếm một nhánh thư mục được chỉ định để



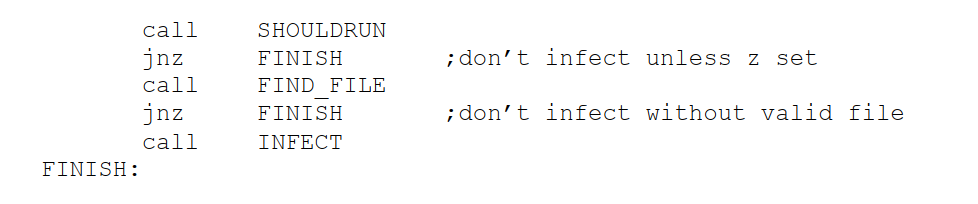
một mức được chỉ định, FIND\_BR. Khi FIND\_BR được gọi, một thư mục đường dẫn được lưu trữ dưới dạng chuỗi ASCII kết thúc null trong biến USEFILE và độ sâu tìm kiếm được chỉ định trong LEVEL. Tại cấp độ thứ ba của thuật toán tìm kiếm, một quy trình tìm kiếm EXE các tập tin (FINDEXE) và hai tìm kiếm thư mục con (FIRSTDIR và NEXTDIR). Quá trình tìm kiếm file EXE sẽ gọi FILE\_OK để xác định xem mỗi tệp mà nó tìm thấy có bị lây nhiễm hay không và nó sẽ dừng mọi thứ khi tìm thấy một tập tin tốt. Logic của việc này trình tự tìm kiếm được minh họa trong Hình 12. Mã cho các thói quen cũng được liệt kê trong Phụ lục B.

**Quy trình chống phát hiện**

Một chiến thuật chống phát hiện khá đơn giản có thể khiến loại virus này mắt người khó xác định vị trí hơn nhiều: Đơn giản là không cho phép các hoạt động tìm kiếm và sao chép sẽ được thực thi mỗi khi virus xâm nhập điều khiển. Một cách dễ dàng để làm điều đó là nhìn vào đồng hồ hệ thống, và xem liệu thời gian tính bằng tích tắc (1 tích tắc = 1/18,2 giây) modulo một số là số không. Nếu đúng như vậy, hãy thực hiện quy trình tìm kiếm và sao chép, nếu không thì chỉ cần chuyển quyền điều khiển cho chương trình chủ. Tính năng chống phát hiện này thói quen sẽ trông như thế này:

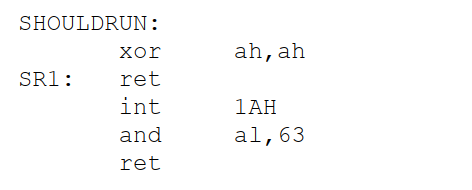


Quy trình này trở lại với tập z khoảng một trong 64 lần. Từ các chương trình thường không được thực thi đồng bộ với đồng hồ bấm giờ, về cơ bản nó sẽ trả về cờ z một cách ngẫu nhiên. Nếu được gọi trong chính thói quen kiểm soát như thế này:

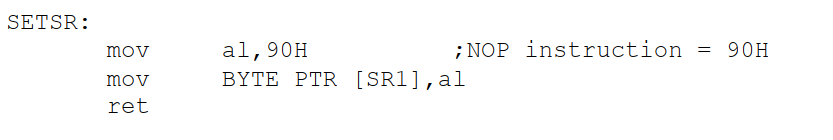


virus sẽ chỉ tấn công một tập tin trong số 64 lần máy chủ chương trình được gọi. Vào những thời điểm khác, virus sẽ vượt qua sự kiểm soát đến máy chủ mà không làm gì cả. Khi làm điều đó, nó sẽ hoàn toàn vô hình ngay cả với con mắt nghi ngờ nhất.

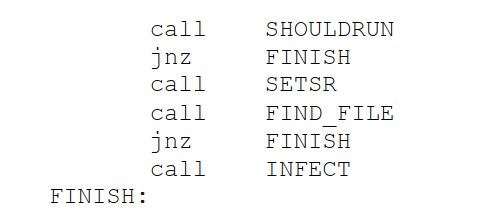
Quy trình SHOULDRUN sẽ gây ra vấn đề nếu bạn muốn đi và lây nhiễm nó vào hệ thống. Bạn có thể phải ngồi đó và chạy chương trình bị nhiễm 50 hoặc 100 lần để virus xâm nhập di chuyển đến một tập tin mới trên hệ thống đó. Điều đó thật khó chịu và có vấn đề nếu bạn muốn đưa nó vào một hệ thống với rủi ro tối thiểu. May mắn thay, một thay đổi nhỏ có thể khắc phục được. Chỉ cần thay đổi SHOULDRUN trông như thế này:



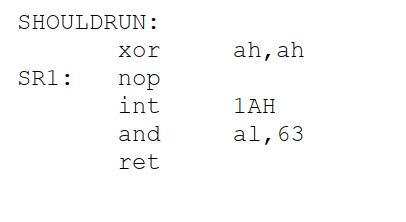
và bao gồm một quy trình khác để sửa đổi quy trình SHOULDRUN,



có thể được kết hợp vào quy trình điều khiển chính như thế này:



Sau khi SETSR được thực thi và trước INFECT, thói quen SHOULDRUN trở thành



vì 90H mà SETSR đặt ở SR1 chỉ là lệnh NOP. Khi INFECT sao chép vi-rút sang một tệp mới, nó sẽ sao chép vi-rút đó bằng phần mở rộng thủ tục SHOULDRUN được sửa đổi. Kết quả là lần đầu tiên virus được thực thi, nó chắc chắn sẽ tìm kiếm một tập tin và lây nhiễm vào nó. Sau đó, nó chuyển sang sơ đồ lây nhiễm 1/64. Bằng cách này, bạn có thể lấy vi-rút đã được tập hợp vào EXE, INTRUDER.EXE và chạy nó và đảm bảo sẽ lây nhiễm thứ gì đó. Sau đó, virus sẽ lây nhiễm vào hệ thống chậm hơn.

Một chiến thuật hữu ích khác mà chúng tôi không sử dụng ở đây là thực hiện lần nhiễm trùng đầu tiên rất hiếm, sau đó thường xuyên hơn. Cái này có thể hữu ích trong việc lây nhiễm vi-rút thông qua BBS, nơi nó có được kiểm tra cẩn thận về hành vi lây nhiễm, và nếu không thấy gì thì đó là đi qua xung quanh. (Đó chỉ là tình huống giả định, xin đừng làm đi!) Trong tình huống như vậy, không ai có thể phát hiện ra virus bằng cách ngồi xuống và chơi với chương trình trong một ngày hoặc hai, ngay cả khi có sẵn một trình kiểm tra vi-rút phức tạp. Tuy nhiên, nếu một rất nhiều người đã chọn một thứ phổ biến và hữu ích (bị nhiễm) chương trình mà họ sử dụng hàng ngày, tất cả họ đều có thể bị nhiễm virus và cuối cùng là lây lan virus.

Sự đánh đổi trong việc hạn chế virus chỉ lây nhiễm vào mọi một trong N lần là nó làm chậm tốc độ lây nhiễm. Điều gì có thể mất một ngày không bị hạn chế có thể mất một tuần, một tháng hoặc thậm chí một năm, tùy thuộc vào tần suất virus được phép sinh sản. Không có quy tắc rõ ràng nào để xác định điều gì là tốt nhất—một cách nhanh chóng tạo ra vi-rút hoặc vi-rút cẩn thận tránh bị chú ý—tất cả phụ thuộc vào những gì bạn đang cố gắng làm với nó.

Một cơ chế chống phát hiện quan trọng khác được kết hợp vào INTRUDER là nó lưu ngày giờ của file đang được bị nhiễm bệnh, cùng với thuộc tính của nó. Sau đó, nó thay đổi thuộc tính tập tin để đọc/ghi, thực hiện các sửa đổi trên tập tin và khôi phục ngày, giờ và thuộc tính ban đầu. Vì vậy, EXE bị nhiễm sẽ không có ngày và giờ lây nhiễm, nhưng ngày ban đầu của nó và thời gian. Sự lây nhiễm không thể được truy ngược lại nguồn gốc của nó bằng cách nghiên cứu ngày của các tập tin bị nhiễm trên hệ thống. Ngoài ra, kể từ bản gốc thuộc tính được khôi phục, bit lưu trữ không bao giờ được thiết lập, vì vậy người dùng thực hiện sao lưu gia tăng không tìm thấy tất cả EXE của anh ấy đang nhận được sao lưu một ngày (một cảnh tượng thực sự kỳ lạ). Là một phần thưởng bổ sung, vi-rút có thể lây nhiễm các tệp hệ thống và chỉ đọc mà không gặp trở ngại nào.

**Chuyển giao quyền điều khiển cho máy chủ**

Bước cuối cùng virus phải thực hiện là chuyển quyền kiểm soát cho chương trình chủ mà không làm rơi bóng. Để làm điều đó, tất cả các thanh ghi nên được thiết lập giống như khi chương trình chủ được được thực thi mà không có virus. Chúng ta đã thảo luận về việc thiết lập cs:ip và ss:sp. Ngoại trừ những điều này, chỉ có thanh ghi rìu được đặt thành giá trị cụ thể của DOS, để cho biết tính hợp lệ của ID ổ đĩa trong FCB có trong PSP. Nếu một mã định danh không hợp lệ (tức là “D:”, khi một hệ thống không có ổ D) nằm trong FCB đầu tiên ở 005C, al được đặt thành FF Hex và nếu định danh hợp lệ, al=0. Tương tự như vậy, ah được đặt thành FF nếu mã định danh trong FCB tại 006C không hợp lệ. Như vậy, rìu có thể đơn giản là được lưu khi virus khởi động và được khôi phục trước khi nó chuyển quyền kiểm soát đến chủ nhà. Các thanh ghi còn lại không được DOS khởi tạo, vì vậy chúng ta không cần quan tâm tới chúng.

Tất nhiên, DTA cũng phải được di chuyển khi virus lây lan lần đầu tiên được kích hoạt và sau đó được khôi phục khi quyền điều khiển được chuyển đến máy chủ. Vì máy chủ có thể cần truy cập các tham số được lưu trữ ở đó, việc di chuyển DTA tạm thời là điều cần thiết vì nó tránh được ghi đè các tham số đó trong quá trình tìm kiếm.

**CẢNH BÁO**

Không giống như virus TIMID, INTRUDER không chứa thông báo rằng nó đang lây nhiễm vào một tập tin. Nó không chứa gì ngoài những thói quen sẽ giúp nó sinh sản. Dù không cố ý phá hoại nhưng nó cực kỳ dễ lây nhiễm và dễ bị bỏ qua. . . và khó thoát khỏi ngay khi nó bắt đầu. Vì vậy, ĐỪNG CHẠY VI-RÚT NÀY, ngoại trừ trong một môi trường được kiểm soát rất cẩn thận. Danh sách ở Phụ lục B chứa mã của virus. Một chương trình định vị, TÌM, cũng được cung cấp, vì vậy nếu bạn chạy vi-rút, bạn sẽ có thể để xem tập tin nào đã bị nhiễm nó.