Cấu 1: Trình bày các khái niệm căn bản: security, availability, confidentiality, integrity, vulnerability, threat, risk,...

Confidentiality: Confidentiality (tính bảo mật) đảm bảo một mức độ bí mật cần thiết trong quá trình xử lý, lưu trữ dữ liệu và ngăn ngừa hành động tiết lộ trái phép.

Availability: Đảm bảo tính sẵn sàng nghĩa là phải đảm bảo được độ tin cậy, luôn luôn truy cập được dữ liệu và tài nguyên khi các cá nhân có thẩm quyền cần đến. Tính sẵn sàng của hệ thống có thể bị ảnh hưởng bởi hỏng hóc thiết bị hoặc lỗi phần mềm.

Integrity: Integrity (tính toàn vẹn) là việc duy trì và đảm bảo độ chính xác cùng với độ tin cậy của thông tin và hệ thống, ngoài ra còn liên quan đến việc ngăn chặn hành động sửa đổi trái phép hệ thống, thông tin. Phần cứng, phần mềm và cơ chế truyền thông cần phải duy trì và xử lý dữ liệu thật chính xác, để đảm bảo dữ liệu đến nơi nhận không bị bất kỳ sự thay đổi nào cả.

Vulnerability: Đề cập đến điểm yếu trong software, hardware, procedural hoặc human mà thông qua đó cung cấp cho Hacker một lối vào, nhằm tiềm nhập vào máy tính hoặc mạng lưới điện toán và truy cập trái phép vào các tài nguyên của tổ chức. Vulnerability là đặc trưng của sự thiếu vắng hoặc lỗ hổng trong cơ chế bảo vệ.

Threat: Mang ý nghĩa là bất kỳ mối nguy hiểm tiềm tàng đối với thông tin hoặc hệ thống. Threat có thể là một ai đó, hoặc một cái gì đó, sẽ thông qua một Vulnerability cụ thể để gây nguy hiểm cho tổ chức hoặc một cá nhân

Risk: Là khả năng xảy ra của một mối nguy hiểm, lợi dụng điểm yếu để gây ảnh hưởng đến tổ chức hoặc năng suất kinh doanh.

Câu 2: Trình bày ưu và nhược điểm của 2 hướng tiếp cận trong việc xây dựng chương trình đảm bảo ATTT trong doanh nghiệp là top-down và bottom-up.

Top-down: Ban hành chính sách, qui trình, Qui định mục tiêu cũng như là kết quả của dự án, Qui định ai chịu trách nhiệm gì

Ưu điểm	Nhược điểm
- Người quản lý hiểu biết toàn bộ cục diện chiến lược an ninh của tổ chức (mối đe dọa, tác động ảnh hưởng, phân bố nhân lực...) - Quản lý an ninh một cách hệ thống xuyên suốt từ trên xuống dưới, xuống đến toàn bộ các nhân viên trong tổ chức. - Phương pháp này đạt được tính toàn diện, đồng bộ và lâu dài.	- Có thể xảy ra thiếu sót trong quá trình xây dựng chính sách bảo mật ban đầu - Chi phí triển khai có thể cao

Bottom-up: Quản trị mạng/hệ thống cố gắng đề nghị các giải pháp để tăng độ an toàn của hệ thống của họ

Ưu điểm	Nhược điểm
- Tận dụng được kiến thức chuyên môn của quản trị hệ thống - Xây dựng hạ tầng thông tin, giải pháp bảo mật trước rồi sau đó mới xây dựng chính sách bảo mật. - Triển khai nhanh hơn - Chi phí thấp	- Phụ thuộc vào quá trình hoạt động của các nhân viên nghiệp vụ trong tổ chức - Người quản lý sẽ không có một sự chuẩn bị, nắm bắt, quản lý về những mối đe dọa tiềm tàng, ảnh hưởng của nó - Các giải pháp công nghệ xây dựng đều không toàn diện, thiếu đâu vá đấy, không mang tính tổng thể đồng bộ - Không được ủng hộ - Khó huy động nhân sự phù hợp

Câu 3: Security through obscurity là gì?

Security through obscurity (STO): Bảo mật bằng cách che dấu.

Security thông qua (hoặc bằng cách) sự che giấu đề cập đến một nguyên tắc trong kỹ thuật an ninh, bằng cách che giấu những thứ như kiến trúc hệ thống, thiết kế nhằm mục đích làm tăng tính an toàn cho chúng. Một hệ thống dựa vào kỹ thuật "security through obscurity" về lý thuyết hoặc trên thực tế chúng có thể đang bị những lỗ hổng bảo mật thực sự,nhưng người quản trị hoặc người thiết kế hệ thống tin rằng sẽ không ai biết được sự tồn tại của lỗ hổng này và Hacker sẽ không thể tìm thấy chúng.

Một hệ thống có thể sử dụng security through obscurity như là một phương pháp Defense-in-the-depth. Bước đầu tiên của Hacker thường là thu thập thông tin, hay còn gọi là Footprinting, nhờ security through obscurity mà chúng ta có làm chậm bước thu thập hoặc đánh lừa, làm nản lòng các Scriptkiddy. Kỹ thuật này được một số các chuyên gia đánh giá là tương phản với lối thiết kế an ninh chuẩn hóa, mặc dù trên thực tế rất nhiều dự án bảo mật áp dụng cả hai chiến lược Security by Design và Security by Obscurity.

Bao gồm những ý tưởng trong security through obscurity như

- Lỗi không bao giờ bị khai thác nếu đối thủ không biết đến chúng
- Mã nguồn đóng được biên dịch an toàn hơn so với mã nguồn mở, bởi vì không ai có thể nhìn thấy mã nguồn của nó
- Chuyển lưu lương HTTP sang port 8088 sẽ an toàn hơn so với 80
- Phát triển các thuật toán mã hóa cá nhân có thể giúp chống lại Crackers

Cho đến ngày nay, Security-by-obscurity là một phương pháp vẫn còn gây tranh cãi, chưa được công nhận trong lĩnh vực an toàn thông tin và mã hóa. Phụ thuộc vào "sự mập mờ" để bảo mật có thể gây ra nguy hiểm.

Câu 4: Ý tưởng chính của hướng tiếp cận defense in depth là gì?

Khi thiết kế một hệ thống phòng thủ, chúng ta phải thiết kế sao cho hệ thống đó có cấu trúc tương tự như…củ hành tây. Kẻ tấn công lột một lớp vỏ bên ngoài, sẽ còn rất nhiều lớp vỏ bên trong bảo vệ cho phần lõi của củ hành, đây chính là khái niệm phòng thủ có chiều sâu, một trong những khái niệm quan trọng nhất khi thảo luận về bảo mật mạng máy tính. Phòng thủ có chiều sâu giúp chúng ta bảo vệ hệ thống mạng của mình bất chấp một hoặc nhiều *lớp* bảo vệ bên ngoài bị xâm hại. Một hệ thống phòng thủ chỉ an toàn khi nào càng đi sâu vào bên trong, kẻ tấn công càng gặp phải nhiều khó khăn, tốn nhiều công sức và dễ bị phát hiện hơn. Một chi tiết cần phải lưu ý là không có bất kì lớp bảo vệ nào đủ sức chống lại mọi loại tấn công, sức mạnh của hệ thống phòng thủ là sự kết hợp sức mạnh của từng lớp bảo vệ, mỗi lớp thực thi nhiệm vụ của riêng mình, nghĩa là ngăn cản và phòng ngừa một loại tấn công cụ thể nhất định.

Có ba yếu tố tạo thành một hệ thống phòng thủ có chiều sâu: network perimeter, internal network, và nhân tố con người.

Câu 5: Phân tích sự ràng buộc giữa việc đảm bảo an toàn thông tin và khả năng truy cập vào hệ thống

Đảm bảo an toàn thông tin và khả năng truy cập vào hệ thống là hai yếu tố luôn mâu thuẫn với nhau, ràng buộc lẫn nhau. Khi một hệ thống được xây dựng với tiêu chí đảm bảo an toàn ở mức cao nhất thì cũng sẽ đem lại khả năng truy cập khó khăn hơn đối với người dùng, nhất là đối với những người dùng thông thường (*Không phải chuyên gia*). Có thể thấy các yếu tố ảnh hưởng đến an toàn thông tin và khả năng truy cập vào hệ thống:

- ***Authentication, Authorization***, ***Encryption***
- ***Discretionary Access Control*** – DAC (*Điều khiển truy cập tuỳ quyền*): người dùng bảo vệ được dữ liệu của mình tránh hạn chế được những truy cập từ những người dùng không được phép.
- ***Mandatory Access Control*** – MAC (*Điều khiển truy cập bắt buộc*): được dùng để bảo vệ một khối lượng dữ liệu lớn cần được bảo mật cao trong một môi trường mà các dữ liệu và người dùng đều có thể được phân loại rõ ràng, hiện thực mô hình bảo mật nhiều mức.
- ***Role-based access control*** – RBAC (*Điều khiển truy cập trên cơ sở vai trò*) là một tiếp cận (*phương pháp*) để hạn chế người dùng hợp pháp truy cập hệ thống.

Xem xét 6 yêu tố trên, ta thấy rằng để đảm bảo an toàn thông tin, cần phải xây dựng các giải pháp khác nhau cho việc truy cập vào hệ thống. Việc áp dụng đồng thời nhiều giải pháp giúp cho hệ thống trở nên an toàn hơn nhưng cũng sẽ tạo ra nhiều hạn chế hơn đối với khả năng truy cập và hiệu suất vận hành toàn hệ thống.

Câu 6: Trình bày 3 loại công cụ dùng để đảm bảo an toàn thông tin là công cụ hành chính, kỹ thuật và vật lý

Công cụ vật lý

- Sử **dụng** các công cụ như: IPD, IDS, Firewall, Cổng ra vào, hàng rào, nhân viên bảo vệ, khóa, thiết bị theo dõi, phát hiện các xâm phạm (các bộ cảm ứng phát hiện chuyển động, thiết bị báo động) nhằm đảm bảo chỉ có những người được cấp quyền mới được vào các khu vực nhạy cảm như: chỗ server CSDL, hệ thống mạng, điện, …

Công cụ kỹ thuật

- **Phần mềm kiểm soát truy cập:** đảm bảo chỉ có những người dùng được cấp quyền mới truy cập được tài nguyên.
- **Phầm mềm duyệt virus:** phát hiện, xác định, cô lập, và tiêu diệt virus.
- **Mã hoá dữ liệu:** bảo vệ dữ liệu khi truyền trên mạng
- **Hệ thống phát hiện xâm nhập:** phát hiện, ngăn ngừa những hoạt động xâm nhập trái phép của người dùng

Công cụ hành chính:

- Đào tạo để nâng cao nhận thức về an toàn thông tin.
- Xây dựng quy trình tuyển dụng để hạn chế rủi ro khi tuyển dụng
- Các thủ tục, quy định nghỉ việc
- Các chính sách và thủ tục bảo đảm an toàn thông tin.
- Thực hiện giám sát

Câu 7: Trình bày các loại rủi ro (về ATTT) thường gặp trong hoạt động của một tổ chức

1. **Rủi ro vật lý**(Physical damage Fire, water, vandalism, power loss, and natural disasters) như: mất điện, cháy, hư hỏng phần cứng, thiên tai làm cho hệ thống bị tê liệt không hoạt động

2. **Tác động người dùng**(Human interaction Accidental or intentional action or inaction that can disrupt productivity) do vô tình hay cố ý người dùng gây ra các tác hại xấu đến hệ thống như: xoá tập tin hệ thống, xoá cơ sở dữ liệu, chỉnh sửa hệ thống, tắt hệ thống

3. **Sự cố phần cứng thiết bị**(Equipment malfunction Failure of systems and peripheral devices) thiết bị bị lỗi, hỏng phần cứng do nhà sản xuất

4. **Tấn công phá hoại từ bên trong và bên ngoài**(Inside and outside attacks Hacking, cracking, and attacking):
Hệ thống bị tấn công từ bên trong do nhân viên bên trong tổ chức gây ra, nhân viên cũ đã nghỉ việc phá hoại.
Hệ thống bị tấn công từ bên ngoài: do bên thứ 3 can thiệp vào hệ thống dưới nhiều dạng tấn công khác nhau làm cho hệ thống ngừng hoạt động, hao tốn tài nguyên, chiếm quyền hệ thống như DDos, Rootkit, Backdoor, virus…

5. Lạm dụng việc chia sẻ dữ liệu, gián điệp, gian lận, đánh cắp thông tin trong hệ thống(Misuse of data Sharing trade secrets, fraud, espionage, and Theft)

6. Mất dữ liệu, lộ thông tin mật do vô tình hay cố ý (Loss of data Intentional or unintentional loss of information through destructive means)

7. Lỗi bảo mật của ứng dụng gây ra về phần cứng lẫn phần mềm, lỗi dữ liệu đầu vào…(Application error Computation errors, input errors, and buffer overflows)

Câu 8: Trình bày kỹ thuật, các bước chính trong việc ước lượng rủi ro đối với 2 hướng tiếp cận quantitative và qualitative

Định lượng (qualitatively): dựa vào khả năng (likelihood)
Định tính (quantitatively): dựa vào xác suất (probability)
Quantitative assessment: dựa vào các giá trị để ước tính

- Tổ chức liệt kê tất cả các khả năng bị tấn công, sau đó ước tính xác suất của mỗi tấn công này với 5 mức độ: non, low, medium, high và very high hoặc sử dụng các cách khác như: A–Z, 0–10, 1–5, và 0–20. Việc này dễ hơn việc đánh giá rủi ro hay đánh giá tài sản một cách chính xác.

Định lượng: (Quantitative):

- Gán rủi ra thông qua con số và giá tiền
- Tất cả rủi ro đều quy ra tiền
- Có thể tiếp cận bằng phương pháp định tính

Định tính: (Quanlitative)

- Dựa trên cảm tính của mỗi người
- Đánh giá qua tỉ lệ (xác xuất)
- Dựa trên kịch bản

Quy trình định lượng:

- Bước 1: Gán giá trị cho tài sản
  Ước tính giá trị mỗi loại tài sản thông qua các câu hỏi:
  - Giá trị của tài sản đó là bao nhiêu?
  - Phí để nó hoạt động?
  - Tiền nó mang lại?
  - Giá trị canh tranh của nó (cái này không hiểu :D)
  - Chi phí để thay mới?
  - Chi phí để mua hoặc phát triển
  - Số tiền bị mất khi tài sản đó không hoạt động
- Bước 2: Ước lượng rủi ro
  Trả lời các câu hỏi
  - Các mối đe dọa về mặt vật lý và chi phí? (Cháy nổ, mất cắp, …)
  - Năng suất giảm như thế nào với các nguyên nhân này và giá là bao nhiêu
  - Chi phí bị mất nếu thông tin bị tiết lộ
  - Chi phí phục hồi khi bị các nguy cơ này
  - Chi phí bị mất nếu các thiết bị không hoạt động
  - What is the single loss expectancy (SLE) for each asset, and
  - each threat? (Cái này bó tay ☹)
- Bước 3: Phân tích các nguy cơ:
  - Thu thập thông tin về rủi ro ở các bộ phận. Xem các giải pháp bảo mật ở quá khứ và hiện tại ra sao
  - Xác định tần suất xuất hiện các rủi ro này. Ví dụ như số lần xuất liện trong 1 năm

- Bước 4: Xem các nguy cơ xuất hiện hàng năm
    - Kết hợp nguy cơ và xác suất
    - Tính số tiền mất hàng năm thông qua 3 bước đầu
    - Chọn các biện pháp khắc phục tương ứng với mỗi rủi ro
    - Phân tích chi phí/lợi ích của mỗi biện pháp
- Bước 5: Giảm thiểu, chuyển, tránh hoặc chấp nhận rủi ro
    - Giảm thiểu rủi ro:
        - Thực hiện các biện pháp/công cụ bảo mật
        - Cải thiện quy trình sẵn có
        - Thay đổi môi trường
    - Chuyển rủi ro: VD mua các gói bảo hiểm
    - Chấp nhận rủi ro: Không cần bỏ chi phí với các rủi ro có tần suất không cao hoặc không có hại gì lớn
    - Tránh rủi ro: Ngừng các hoạt động có thể gây ra rủi ro

Quy trình định tính:

Đánh giá mức độ nghiêm trọng của rủi ro thông qua: Judgment, Best practices, Intuition, Experience

Một số công cụ:

- Delphi
- Động não
- Viết kịch bản (Storyboarding)
- Khảo sát nhóm
- Khảo sát (survey)
- Đặt câu hỏi
- Phỏng vấn từng người

Nhược điểm của định tính:

- Chủ quan
- Không đưa ra được giá tiền
- Không có chuẩn chung

Nhược điểm của định lượng:

- Tính toán phức tạp
- Khó khăn khi không có công cụ
- Trải qua nhiều bước

Câu 9: Trình bày cơ chế hoạt động của các loại lỗi SQL injection, parameter injection, broken authentication, XSS.

SQL injection: lợi dụng các câu truy vấn xuống CSDL, kẻ tấn công sẽ chèn thêm các tham số làm cho câu truy vấn đó sai lệch, nhằm mục đích thay đổi dữ liệu hay truy cập dữ liệu trái phép.

Có 2 loại:

Numeric SQL Injection: chèn các tham số dạng number vào câu truy vấn. Áp dụng với những câu truy vấn kiểu số.

VD: SELECT * FROM weather_data WHERE station = ?

Với ? là tham số, theo bình thường chúng ta chỉ cần điền vào station id = 100 là được, nếu kẻ xấu truyền 100 OR 1=1, câu query này luôn đúng, nên có thể làm cho hệ thống bạn bị đánh cắp thông tin.

String SQL Injection: chèn các tham số kiểu chuỗi vào câu truy vấn. Áp dụng với các câu truy vấn kiểu chuỗi.

VD: giống như numberic SQL injection, thay vì dùng dạng số, ta dùng dạng chuỗi:

SELECT * FROM employee WHERE name = 'thanh' OR '1'='1';

Ứng dụng:

- ✓ Dùng để xem tất cả dữ liệu trong hệ thống
- ✓ By pass authentication
- ✓ By pass Authorization
- ✓ Execute query: CRUD, create DB backdoor, stored procedures

Parameter injection: lợi dụng điểm yếu của hệ điều hành hiện tại, thực hiện 1 số lệnh liên quan đến shell như: ls, cp, rm.... thông qua các tham số truyền vào.

Broken authentication: vượt qua các chứng thực của hệ thống một cách không tường minh. Có nghĩa là kẻ tấn công không cần biết username, password, hay key...

Lợi dụng vào các điểm yếu như: Cookies, Session, Buffer Overflows..

XSS: dụ dỗ user click/submit vào đường link mà attacker tạo để lấy thông tin của user đó.

Có 2 dạng:

- ✓ Stored: chèn thêm đoạn mã ở các vùng nhập liệu: search, comment, message, forum log, sau đó đoạn mã này sẽ được lưu xuống server. Người khác vô tình hay bị lừa click vô link, sau đó kẻ tấn công sẽ lấy được thông tin của người lick.
- ✓ Reflected: bằng cách nào đó, kẻ tấn công chèn được các đoạn mã vào kết quả mà server trả về: search, error, success message, hay là 1 input form..Sau đó người dùng nhập thông tin vào các input này, bấm submit -> hacker lấy được thông tin.

Câu 10: Mã hoá: Phân biệt giữa mã hoá đối xứng và bất đối xứng. So sánh các ưu và nhược điểm của mã hoá đối xứng và bất đối xứng

	Mã hóa đối xứng	Mã hóa bất đối xứng
Định Nghĩa	Là thuật toán các mã hóa trong đó việc mã hóa và giải mã đều dùng chung cho 1 khóa (secret key). Thuật toán đối xứng có thể được chia ra làm hai thể loại, mật mã luồng (*stream ciphers*) và mật mã khối (*block ciphers*). Mật mã luồng mã hóa từng bit của thông điệp trong khi mật mã khối gộp một số bit lại và mật mã hóa chúng như một đơn vị. Cỡ khối được dùng thường là các khối 64 bit. Một số ví dụ các thuật toán đối xứng nổi tiếng và khá được tôn trọng bao gồm Twofish, Serpent, AES (còn được gọi là Rijndael), Blowfish, CAST5, RC4, Tam phần DES (*Triple DES*), và IDEA (*International Data Encryption Algorithm – Thuật toán mật mã hóa dữ liệu quốc tế*).	Là thuật toán trong đó việc mã hóa và giãi mã dùng hai khóa khác nhau là pulic key(khóa công khai hay khóa công cộng) và private key (khóa riêng). -Nếu dùng public key để mã hóa thì private key sẽ dùng để giải mã và ngược lại
Ưu điểm	**Tốc độ** Các thuật toán đối xứng nói chung đòi hỏi công suất tính toán ít hơn các thuật toán khóa bất đối xứng (*asymmetric key algorithms*). Trên thực tế, một thuật toán khóa bất đối xứng có khối lượng tính toán nhiều hơn gấp hằng trăm, hằng ngàn lần một thuật toán khóa đối xứng (*symmetric key algorithm*) có chất lượng tương đương. Các thuật toán đối xứng thường không được sử dụng độc lập. Trong thiết kế của các hệ thống mật mã hiện đại, cả hai thuật toán bất đối xứng (*asymmetric*) (dùng chìa khóa công khai) và thuật toán đối xứng được sử dụng phối hợp để tận dụng các ưu điểm của cả hai. Những hệ thống sử dụng cả hai thuật toán bao gồm những cái như SSL (*Secure Sockets Layer*), PGP (*Pretty Good Privacy*)	**An toàn** Về khía cạnh an toàn, các thuật toán mật mã hóa bất đối xứng cũng không khác nhiều với các thuật toán mã hóa đối xứng. Có những thuật toán được dùng rộng rãi, có thuật toán chủ yếu trên lý thuyết; có thuật toán vẫn được xem là an toàn, có thuật toán đã bị phá vỡ… Cũng cần lưu ý là những thuật toán được dùng rộng rãi không phải lúc nào cũng đảm bảo an toàn. Một số thuật toán có những chứng minh về độ an toàn với những tiêu chuẩn khác nhau. Nhiều chứng minh gắn việc phá vỡ thuật toán với những bài toán nổi tiếng vẫn được cho là không có lời giải trong thời gian đa thức . Nhìn chung, chưa có thuật toán nào

	và GPG (*GNU Privacy Guard*) v.v. Các thuật toán chìa khóa bất đối xứng được sử dụng để phân phối chìa khóa mật cho thuật toán đối xứng có tốc độ cao hơn.	được chứng minh là an toàn tuyệt đối (như hệ thống mật mã sử dụng một lần). Vì vậy, cũng giống như tất cả các thuật toán mật mã nói chung, các thuật toán mã hóa khóa công khai cần phải được sử dụng một cách thận trọng.
Nhược điểm	**Số lượng khoá nhiều, không an toàn, khó vận hành** Hạn chế của các thuật toán khóa đối xứng bắt nguồn từ yêu cầu về *sự phân hưởng chìa khóa bí mật*, mỗi bên phải có một bản sao của chìa. Do khả năng các chìa khóa có thể bị phát hiện bởi đối thủ mật mã, chúng thường phải được bảo an trong khi phân phối và trong khi dùng. Hậu quả của yêu cầu về việc lựa chọn, phân phối và lưu trữ các chìa khóa một cách không có lỗi, không bị mất mát là một việc làm khó khăn, khó có thể đạt được một cách đáng tin cậy. Để đảm bảo giao thông liên lạc an toàn cho tất cả mọi người trong một nhóm gồm n người, tổng số lượng chìa khóa cần phải có là $\frac{n(n-1)}{2}$ Các thuật toán **khóa đối xứng** không thể dùng cho mục đích xác thực (*authentication*) hay mục đích chống thoái thác (*non-repudiation*) được.	**Xử lý nhiều, tốc độ chậm** Tồn tại khả năng một người nào đó có thể tìm ra được khóa bí mật. -Khả năng bị tấn công dạng **kẻ tấn công đứng giữa** (man in the middle attack): kẻ tấn công lợi dụng việc phân phối khóa công khai để thay đổi khóa công khai. Sau khi đã giả mạo được khóa công khai, kẻ tấn công đứng ở giữa 2 bên để nhận các gói tin, giải mã rồi lại mã hóa với khóa đúng và gửi đến nơi nhận để tránh bị phát hiện. -Tốc độ mã hoá chậm để đạt được độ an toàn tương đương đòi hỏi khối lượng tính toán nhiều hơn đáng kể so với thuật toán mật mã hóa đối xứng.

Vì thế trong thực tế hai dạng thuật toán này thường được dùng bổ sung cho nhau để đạt hiệu quả cao. Hiện nay người ta phổ biến dùng các thuật toán bất đối xứng có tốc độ chậm hơn để phân phối chìa khóa đối xứng khi một phiên giao dịch bắt đầu, sau đó các thuật toán khóa đối xứng tiếp quản phần còn lại . Vấn đề về bảo quản sự phân phối chìa khóa một cách đáng tin cậy cũng tồn tại ở tầng đối xứng, song ở một điểm nào đấy, người ta có thể kiểm soát chúng dễ dàng hơn.khóa bất đối xứng. Sau đó 2 bên trao đổi thông tin bí mật bằng hệ thống mã hóa đối xứng trong suốt phiên giao dịch.

Câu 11: Giải thích tấn công mã hoá thay thế (substitution) bằng kỹ thuật thống kê tần suất xuất hiện ký tự (English letter frequency)

Ý tưởng:

Mã hoá thay thế (substitution) là một giải thuật mã hóa dựa trên bảng chữ cái Alphabet để chuyển đổi từ **Plain text**(nội dung chưa mã hóa) thành **Cypher text**(nội dung đã được mã hóa)
Kỹ thuật thống kê tần suất xuất hiện ký tự (English letter frequency): là kỹ thuật tấn công mã hóa thay thế bằng cách nhìn vào tần suất xuất hiện của các kỹ tự trong **Cypher text** mà đoán ra nội dung của **Plain text** ban đầu như sau:
Trong tiếng anh (chỉ đúng cho tiếng anh tiếng việt thì có quy luật khác) những chữ cái khác nhau chiếm tần xuất xuất hiện khác nhau, chúng được chia thành 5 nhóm với tần suất xuất hiện giảm dần:

I:	e
II:	t,a,o,i,n,s,h,r
III:	d,l
VI:	c,u,m,w,f,g,y,p,b
V:	v,k,j,x,q,z

Qua đó, nếu nhìn vào cypher text(nội dung đã được mã hóa) mà thấy những ký tự nào xuất hiện nhiều(ví dụ chữ Z) thì có thể suy đoán bảng mã thay thế của mình là: **e -> Z**, và làm tương tự cho đến khi đoán được **Plain text**(nội dung có nghĩa).

Nguồn:
http://soict.hust.edu.vn/~vannk/AntoanThongtin/GTATTT/GTATTT_Chuong1.pdf

Chi tiết

1.2.1 Mật mã một bảng thế (Monoalphabetic cipher)

Ở đây thuật toán dựa trên phép hoán vị trong một bảng chữ cái alphabet.

Ví dụ 1.1. Một cipher dựa trên một bảng hoán vị của tiếng Anh như sau

a	b	c	d	e	...	x	y	z
F	G	N	T	A	...	K	P	L

Qua bảng biến đổi có thể thấy a➔ F, b➔G … Qua đó sẽ có

Plaintext:	a	bad	day
➔			
Ciphertext:	F	GFT	TFP

Như vậy khoá trong một cipher loại này là một bảng hoán vị (A →F, b→G, ...,

z→L) như trên, hoặc biểu diễn ngắn gọn hơn là bằngdòng thứ hai của phép biến đổi này, tức là FGNT..PL. Dòng thứ nhất của bảng biến đổi này là bảng chữ cái gốc, vì nó là cố định nên không được tính tới trong khoá. Dòng thứ hai, được gọi là bảng thay thế (substitution alphabet).

Chú ý rằng không nhất thiết phải dùng một bảng chữ cái mà ta có thể dùng bất cứ một thứ bảng ký hiệu nào đó.

Ví dụ 1.2. Ở đây bảng chữ bản rõ, *plaintext alphabet*, là một tập hợp của các xâu nhị phân với độ dài là 3.

Bảng biến đổi:

p.text	000	001	010	011	100	101	110	111
c.text	101	111	000	110	010	100	001	011

Do đó xâu nhị phân plaintext 100101111 sẽ được mã hoá thành 010100011.

Để giải mã một bản rõ nhận được từ thuật toán mật mã trên, người có bản mã ciphertext cần biết khóa, do đó yêu cầu một giao thức về trao khoá. Đơn giản nhất có thể thực hiện là người gửi tin ghi khoá ra đĩa và chuyển đĩa cho người nhận. Rõ ràng cách làm này đơn giản nhưng thực tế không an toàn. Trong thực tế người ta sử dụng nhiều giao thức phức tạp và tinh vi hơn.

Nếu như kẻ thù không biết được khoá thì liệu chúng có thể đoán được không ? Hiển nhiên là điều đó phụ thuộc vào số lượng khoá có thể có (độ lớn của không gian khoá có thể có). Nếu kích thước của bảng alphabet là N thì số khoá có thể là N! =N(N-1)...1 và được tính xấp xỉ theo công thức:

$$N! \approx (2\pi n)^{1/2}(n/e)^n$$

Cho N=26, ta có $N!=26!\approx 9^{26}$.

Chú ý rằng, số lượng bit được chuyển mật này được gọi là chiều dài của khoá.

Ví dụ 1.3. Chiều dài khoá của một cipher loại đang xét là 26*5=130 bits, chính là số lượng bit tin cần dùng để chuyển đi dòng thứ hai trong bảng chuyển vị trên. (Dòng thứ nhất đã được ngầm định là ABC..XYZ, nên không cần chuyển).

---

a b	c	d	e	...	x	y	z	
A	B	C	D	E	...	X	Z	Y

Chú ý: Không phải tất cả các cipher như trên là che giấu được nội dung của thông tin.

Ví dụ 1.4: Sau đây là một cipher hầu như không làm thay đổi plaintext.

Mật mã cộng (Additive cipher) - Mật mã Xeda (Ceasar)

Mật mã cộng (Additive cipher) là một mật mã một bảng thế đặc biệt trong đó, phép biến đổi mã được biểu diễn thông qua phép cộng đồng dư như sau. Giả sử ta gán các giá trị từ A-Z với các số 1-25,0. Thế thì một chữ plaintext X có thể mã thành ciphertext Y theo công thức:

$$Y = X \oplus Z,$$

trong đó Z là giá trị của khoá, $\oplus$ là ký hiệu phép cộng đồng dư modulo 26.

Ví dụ 1.5 Xét mật mã một bảng thế sau đây:

a	b	c	d	e	...	x	y	z
D	E	F	G	H	...	A	B	C

Đây chính là mật mã Ceasar đã giới thiệu từ đầu chương, trong đó giá trị khóa là Z=3: D=a $\oplus$3, E=b $\oplus$3,... A=x $\oplus$3, B=y $\oplus$3, C=z $\oplus$3

Rõ ràng số lượng khoá có thể dùng được chỉ là 25 và số lượng bít cần thiết cho việc chuyển khoá là 5 ($2^4 < 25 < 2^5$). Có thể thấy rằng mật mã cộng có một không gian khoá rất nhỏ, do đó phép tìm kiếm vét cạn đương nhiên là khả thi. Trong phép tấn công này, địch thủ chỉ cần thử tất cả các khoá có thể (1-25) để thử giải mã và dễ dàng phát hiện ra khoá đúng khi giải ra một thông tin có nghĩa. Vì phép tìm kiếm này không cần sử dụng các quan sát tinh tế mà chỉ đơn giản là thử hết các khả năng, dựa vào sức mạnh tính toán của kẻ tấn công, nên nó cũng còn được biết với cái tên *tấn công vũ lực* (*brute force attack*)

Mật mã nhân tính (multiplicative cipher)

Bảng thế cũng có thể được xây dựng từ phép nhân đồng dư của chữ cái trong bảng gốc với giá trị của khóa:

$$Y=X \otimes Z$$

Trong đó $\otimes$ là phép nhân đồng dư với modul 26.

Tuy nhiên chú ý rằng không phải tất cả các giá trị từ 1-25 đều có thể là khoá mà chỉ các giá trị nguyên tố cùng nhau với 26, tức là các số lẻ trừ 13. Do đó chỉ có 12 khoá cả thảy mà thôi.

Ví dụ 1.6. Nếu ta dùng khóa Z=2

$2 \otimes 1 = 2 \bmod 26$ tức là b $\rightarrow$ c.

nhưng $2 \otimes 14 = 2 \bmod 26$ tức là o $\rightarrow$ c

Rõ ràng khoá 2 không thoả mãn, vì không tạo ra ánh xạ 1-1 từ bảng chữ gốc sang bảng thay thế. Sự kiện đồng thời có b$\rightarrow$c, và o$\rightarrow$c sẽ làm cho ta không thể giải mã ciphertext *c*.

Để tăng số lượng khoá có thể, người ta có thể kết hợp cả *additive cipher* và

multiplicative cipher để tạo ra *afine cipher*:

$$Y = \alpha \otimes X \oplus Z$$

$$X, Y, Z \in \{ 0,1,2,3,...25\}$$

$$\alpha \in \{ 1,3,5,7,9,11,15,17,19,21,23,25\}$$

Qua những khảo sát trên ta có thể dễ dàng thấy các dạng đặc biệt của mật mã bảng thế (trong đó phép biến đổi mật mã là một hàm toán học đơn giản) là không an toàn ngay cả với tấn công tìm kiếm vét cạn. Tuy nhiên mật mã một bản thế tổng quát, sử dụng một hoán vị bất kỳ trên bảng chữ cái gốc, có không gian khóa là thường là đủ lớn để chống lại bất kỳ kẻ địch nào (ngay cả trong thế giới hiện đại) chỉ dùng tấn công vét cạn -- cụ thể là với bảng chữ cái tiếng Anh (26 chữ), số lượng hoán vị có thể (tức số lượng khóa cần vét cạn) sẽ lên tới $26! \approx 9^{26}$!

Trong thời kỳ thiên nhiên kỷ đầu tiên (trước năm 1000), mật mã một bảng thế được coi là không thể phá được. Tuy nhiên sau đó, các nhà nghiên cứu thời đó đã dần dần tìm ra phương pháp phá giải tốt hơn việc thử vét cạn không gian khóa; phương pháp này dựa trên những quan sát mang tính thông kê, chẳng hạn về sự xuất hiện không đồng đều của các chữ cái trong ngôn ngữ tự nhiên.

1.2.2 Phân tích giải mã theo phương pháp thống kê (Statistical cryptanalysis)

Dễ dàng quan sát một đặc tính của ngôn ngữ tự nhiên là sự xuất hiện (tần xuất) không đều của các chữ cái được dùng khi diễn đạt một ngôn ngữ.

Ví dụ 1.7 Hãy theo dõi một đoạn văn bản sau đây trong tiếng Anh.

THIS IS A PROPER SAMPLE FOR ENGLISH TEXT. THE FREQUENCIES OF LETTERS IN THIS SAMPLE IS NOT UNIFORM AND VARY FOR DIFFERENT CHARACTERS. IN GENERAL THE MOST FREQUENT LETTER IS FOLLOWED BY A SECOND GROUP. IF WE TAKE A CLOSER LOOK WE WILL NOTICE THAT FOR BIGRAMS AND TRIGRAMS THE NONUNIFORM IS EVEN MORE.

Ở đây ta dễ dàng thấy tần suất xuất hiện của chữ cái X và A: f_x=1 và f_A=15.

Khái quát hơn, trong tiếng Anh căn cứ vào tần xuất xuất hiện của các chữ cái trong văn viết, ta có thể chia 26 chữ cái thành 5 nhóm theo thứ tự từ hay dùng hơn đến ít dùng hơn như sau:

I: e

II: t,a,o,i,n,s,h,r

III: d,l

VI: c,u,m,w,f,g,y,p,b

V: v,k,j,x,q,z

Với những quan sát tương tự áp dụng cho các cặp (bigrams) hay bộ ba chữ (trigram), người ta thấy tần xuất cao nhất rơi vào các cụm phổ biến sau:

Th, he, in, an, re, ed, on, es, st, en at, to

The, ing, and, hex, ent, tha, nth, was eth, for, dth.

Chú ý: Những quan sát này được phản ánh trên chính đoạn văn bản ví dụ tiếng Anh ở trên. Những quan sát này chỉ đúng với tiếng Anh và như vậy tiếng Việt của chúng ta sẽ có qui luật khác.

Sau khi đã có các quan sát như trên, người ta có thể dùng phương pháp đoán chữ và giải mã dựa trên việc thống kê tần xuất xuất hiện các chữ cái trên mã và so sánh với bảng thống kê quan sát của plaintext. Ví dụ sau đây sẽ minh họa cụ thể phương pháp này

Ví dụ 1.8 Giả sử ta thu được một đoạn mã một bảng thế như sau và cần phải giải tìm khóa của nó.

YKHLBA JCZ SVIJ JZB TZVHI JCZ VHJ DR IZXKHLBA VSS RDHEI DR YVJV LBXSKYLBA YLALJVS IFZZXC CVI LEFHDNZY EVBLRDSY JCZ FHLEVHT HZVIDB RDH JCLI CVI WZZB JCZ VYNZBJ DR ELXHDZSZXJHDBLXI JCZ XDEFSZQLJT DR JCZ RKBXJLDBI JCVJ XVB BDP WZ FZHRDHEZY WT JCZ EVXCLBZ CVI HLIZB YHVEVJLXVSST VI V HXXIKSJ DR JCLI HZXZBJ YZNZXDFEZBJ LB JZXCBDSDAT EVBT DR JCZ XLFCZH ITIJZEIJCVJ PZHZ DBXZ XDBILYXHZYIZKHZ VHZBDP WHZVMVWSZ.

Đoạn mã trên bao gồm 338 chữ, thống kế tần xuất như sau:

Letter:	A	B	C	D	E	F	G
Frequency:	5	24	19	23	12	7	0
Letter:	H	I	J	K	L	M	N
Frequency:	24	21	29	6	21	1	3
Letter:	O	P	Q	R	S	T	U
Frequency:	0	3	1	11	14	8	0
Letter:	V	W	X	Y	Z		
Frequency:	27	5	17	12	45		

Quan sát Z là chữ mã có tần suất lớn hơn hẳn các chữ cái còn lại nên rút ra:

$e \Rightarrow Z$ (tức là bản rõ của mã Z phải là e)

Quan sát những chữ mã có tần suất cao tiếp theo $f_j = 29$, $fv = 27$

Đồng thời chú ý đến bộ ba *jcz* có tần suất cao, dễ thấy

$$f_{jcz} = 8 \rightarrow t \Rightarrow J,\ h \Rightarrow C$$

(suy luận *jcz* chính là từ bản rõ *the*)

Ngoài ra tiếp tục quan sát ta sẽ thấy một số phát hiện dễ nhận:

$$a \Rightarrow V \text{ (đứng riêng, mạo từ a)}$$

Liệt kê nhóm II gồm các chữ mã có tần suất xuất hiện cao (nhóm 1 là chỉ gồm Z)

J,V,B,H,D,I,L,C ứng với bản rõ của nhóm II: {t,a,o,i,n,s,h,r}

t,a h

Quan sát thấy có một cụm 3 là JZB ($\Rightarrow$ teB), ta sẽ tìm nốt bản rõ của B bằng cách đơn giản sau: thay thế các khả năng nhóm 2 của B vào cụm này:

Teo

Ten

JZB = te ? *ter → n ⇒ B*

The

Tes

Tương tự ta thực hiện một số quan sát và suy đoán khác

VI = a ? as

an → s ⇒ I (n đã có B rồi)

VHZ = a ?e ate

are → r ⇒ H (t đã có J rồi)

JCLI = th?s → i ⇒ L,

Cuối cùng còn lại trong nhóm II: *o ⇒ D*

A	*b*	*C*	*d*	*e*	*F*	*g*	*h*	*i*	*j*
V				*Z*			*C*	*L*	
K	*l*	*M*	*n*	*o*	*P*	*q*	*r*	*s*	*t*
			B	*D*			*H*	*I*	*J*
U	*v*	*W*	*x*	*y*	*z*				

Tiếp tục phân tích nhờ các cụm từ (bản mã) tương đối ngắn:

DBXZ = on?e → c ⇒ X WZZB = ?een =

→ b ⇒ W YVJV = ?ata → d ⇒ Y

Tuy nhiên cũng có trường hợp không chắc chắn:

DR = o ?

on: loại vì n ⇒ B rồi

of:

or: loại vì r⇒H rồi

ox :

Nhưng chưa rõ ràng: f, x ⇒ R Tiếp tục một số luận đoán:

WT = b ? → y ⇒ T BDP = no ? →

w ⇒ P Bây giờ từ đầu tiên sẽ là

YKHLBA = d-rin-

→ u ⇒ K, g ⇒ A

Rõ ràng qua ví dụ trên ta thấy hệ mật mã một bảng thế có thể khá dễ dàng bị phá khi nó vẫn tiếp tục "bảo tồn" trong bản mã những qui luật ngôn ngữ trong bản rõ.

Câu 12: Phân biệt block cipher và stream cipher.

Block ciphers: mã hoá khối – trong đó từng khối dữ liệu trong văn bản ban đầu được thay thế bằng một khối dữ liệu khác có cùng độ dài. Độ dài mỗi khối gọi là block size, thường được tính bằng đơn vị bit. Ví dụ thuật toán 3-Way có kích thước khối bằng 96 bit.

- Stream ciphers: mã hoá dòng – trong đó dữ liệu đầu vào được mã hóa từng bit một. Các thuật toán dòng có tốc độ nhanh hơn các thuật toán khối, được dùng khi khối lượng dữ liệu cần mã hóa chưa được biết trước, ví dụ trong kết nối không dây. Có thể coi thuật toán dòng là thuật toán khối với kích thước mỗi khối là 1 bit.

Câu 13: Chữ ký số là gì? Cách thức thực hiện

Các hệ thống mật mã hóa khóa công khai cho phép mật mã hóa văn bản với khóa bí mật mà chỉ có người chủ của khóa biết. Để sử dụng chữ ký số thì văn bản cần phải được mã hóa bằng hàm băm Hash(văn bản được "băm" ra thành chuỗi, thường có độ dài cố định và ngắn hơn văn bản) sau đó dùng khóa bí mật của người chủ khóa để mã hóa, khi đó ta được chữ ký số. Khi cần kiểm tra, bên nhận giải mã (với khóa công khai) để lấy lại chuỗi gốc (được sinh ra qua hàm băm ban đầu) và kiểm tra với hàm băm của văn bản nhận được. Nếu 2 giá trị (chuỗi) này khớp nhau thì bên nhận có thể tin tưởng rằng văn bản xuất phát từ người sở hữu khóa bí mật. Tất nhiên là chúng ta không thể đảm bảo 100% là văn bản không bị giả mạo vì hệ thống vẫn có thể bị phá vỡ.

Câu 14: Chứng minh OTP (One Time Pad) là thuật toán có tính chất perfect secrecy (theo định nghĩa của Shannon năm 1949)

One-time pad: "mật mã sử dụng một lần". Tôi thấy cũng chưa ổn lắm. Có lẽ là "mật mã khóa sử dụng một lần" thì rõ nghĩa hơn.
Theo cuốn sách The Code Book của tác giả Simon Singh. One-Time Pad có nghĩa là mật mã sổ tay dùng một lần. Mật mã này dùng một sổ tay dày hàng trăm trang, với các trang có các ký tự ngẫu nhiên hoàn toàn làm sổ mã. Mỗi khi mã hóa một bức thư nào, họ(người tạo mã) sẽ lấy một số lượng chữ cái = số ký tự của văn bản cần gửi làm khóa mã cho mật mã hình vuông Vigenère.
Việc này khiến loại mật mã này được coi là Le chiffre indéchiffrable-mật mã không thể phá nổi, tên này trước kia thuộc về mật mã hình vuông Vigenère, vì nó tránh được tất cả mọi yếu điểm của các loại mật mã: số lượng khóa mã rất lớn, không bị quay vòng khóa mã như mật mã Vigenère gốc=>phép thử Kasiski vô hiệu, đồng thời không thể phân tích tần xuất như mật mã sơ cấp, và vì sự ngẫu nhiên hoàn toàn của khóa mã, nên không thể phân biệt được bức thư thật với hàng trăm bức thư giả khác.
Tuy vô địch về mặt bảo mật, nhưng mật mã này quá đắt(chi phí cao), vì khó có thể tạo ra được số lượng lớn khóa mã ngẫu nhiên hoàn toàn, nên cũng ít được dùng, chỉ có một số thông tin liên lạc siêu quan trọng mới sử dụng mà thôi.

Gilbert Vernam là một kỹ sư làm việc tại phòng nghiên cứu AT&T Bell. Năm 1971, ông đã phát minh ra *stream cipher* và đồng phát minh ra thuật toán ***one-time pad*** (OTP). Về lý thuyết, OTP là thuật toán duy nhất chứng minh được là không thể *crack,* ngay cả với tài nguyên vô tận (do đó có thể chống lại*brute-force attack*). Để đạt mức độ bảo mật, tất cả những điều kiện sau phải được thỏa mãn:
– Độ dài của chìa khóa phải đúng bằng độ dài văn bản cần mã hóa.
– Chìa khóa chỉ được dùng một lần.
– Chìa khóa phải là một số ngẫu nhiên thực.

OTP là phương pháp mã hoá tuyệt đối an toàn nếu được sử dụng đúng cách, và là phương pháp tuyệt đối an toàn duy nhất cho đến thời điểm hiện tại. Văn bản được mã hoá với OTP không cho biết bất kỳ thông tin gì về văn bản gốc, ngoại trừ độ dài. Với một văn bản đã mã hoá cho trước, chúng ta có thể nghĩ ra các chuỗi khoá để "giải mã" nó về bất kỳ văn bản nào chúng ta muốn! Các phương pháp mã hoá mới sau này như DES (Data Encryption Standard), AES (Advanced Encryption Standard), PGP (Pretty Good Privacy), PKI (Public Key Infastructure)… tuy tiện dụng và có nhiều ưu điểm khác, nhưng về mặt lý thuyết không phải là không phá được. Nhưng trong sử dụng thực tế, có những lý do sau khiến OTP trở nên không an toàn:

- Chuỗi khóa OTP không thực sự ngẫu nhiên (các nhân viên thư ký của KGB tạo ra OTP bằng cách gõ ngẫu nhiên lên máy đánh chữ, nhưng xu hướng gõ phím của tay người vẫn có những pattern nhất định).

- Việc cất giữ và tiêu huỹ OTP có quá nhiều yếu tố rủi ro (đã có tình huống CIA giải được mã nhờ một cuốn sổ OTP đã bị đốt nhưng chưa cháy hết).
- Mỗi trang OTP chỉ được dùng một lần (đã có lúc trong tình hình khẩn cấp, nhân viên KGB bất cẩn dùng một trang OTP cho nhiều lần mã hoá, dẫn đến việc CIA giải được khoảng 1% trong số những thông điệp gửi bởi KGB trong những năm 1945 ~ 1950).

Điểm yếu nhất của OTP nằm trong quá trình trao đổi khoá (key exchange), đó là một trong những lý do hình thành phương pháp public key rất tiện dụng sau này. Đến bây giờ, khi những phương tiện mã hoá và truyền thông đã quá hiện đại, người ta vẫn còn tiếp tục dùng OTP cho những kênh thông tin thuộc loại top secret (như đường dây hotline Washington DC – Moscow, liên lạc với tàu ngầm…) vì tính tuyệt đối an toàn đã được chứng minh lý thuyết của nó. Có thể kiểm chứng dấu vết của việc sử dụng OTP trong thực tế:

Có một cách sử dụng OTP đặc biệt gọi là chia xẻ bí mật (secret splitting), sau khi mã hoá, văn bản gốc bị huỹ thay vì khoá, sau đó khoá và văn bản mã hoá được đưa cho hai người khác nhau cất giữ. Chỉ khi hai người này cũng đồng ý nối hai "khoá" lại với nhau thì mới giải mã ra được văn bản gốc. Tương tự, có thể chia xẻ bí mật cho 3, 4,… người bằng cách sử dụng 2, 3,… khoá. Đây là cách bảo vệ các tài nguyên đặc biệt quan trọng, trách nhiệm bảo vệ đó được chia xẻ cho nhiều người, tuy nhiên lưu ý rằng nếu chỉ một phần của bí mật bị mất đi, thì bí mật đó cũng sẽ mất đi vĩnh viễn.

15. Định nghĩa các loại mã độc thông dụng: worm, virus, trojan horse, rootkit

Worm: Một loại virut máy tính có khả năng tự nhân bản. Worms lây lan chủ yếu là do lỗ hổng bảo mật trong HĐH. Worm có đủ khả năng để làm thiệt hại nghiêm trọng cho toàn thể mạng lưới, trong khi một virus thông thường chỉ thường nhắm đến các tập tin trên máy bị nhiễm.

Virus: Là một chương trình mà có thể lây lan (lặp lại) từ một máy tính khác. Virus thường phải được đưa thẳng vào một tập tin thực thi để chạy. Khi tập tin thực thi bị nhiễm được khởi chạy, nó có thể sẽ lây lan sang các file thực thi khác với tốc độ rất nhanh. Virus lây lan thường đòi hỏi một số can thiệp của người dùng. Virus có nhiều cách rất khéo léo để chèn mình vào các file thực thi.

Trojan horse: là một chương trình phần mềm độc hại mà không cố gắng để tự tái tạo, thay vào đó nó sẽ được cài đặt vào hệ thống của người dùng bằng cách giả vờ là một chương trình phần mềm hợp pháp. Khi một Trojan Horse được cài đặt trên máy tính của người dùng, nó sẽ không cố gắng để gài chính nó vào một tập tin như virus, nhưng thay vào đó nó sẽ cho phép các hacker hoạt động để điều khiển máy tính của người dùng từ xa

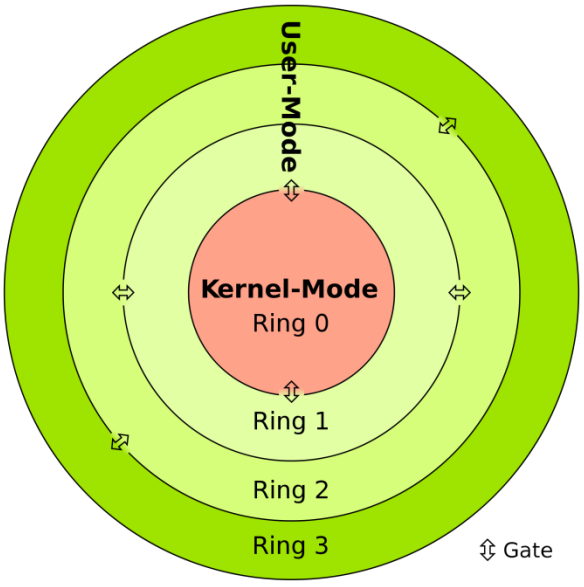
Rootkit: Là loại phần mềm độc hại rất khó để phát hiện vì nó cố gắng để tự ẩn mình trốn thoát người sử dụng, HĐH và các chương trình Antivirus/Anti- malware. Chúng có thể được cài đặt trong nhiều cách, trong đó có phương án khai thác một lỗ hổng trong HĐH hoặc bằng cách tiếp cận quản trị viên máy tính

Câu 16: Giải thích cơ chế hoạt động của Rootkit

Khái niệm rootkit

Khái niệm rootkit được sử dụng để mô tả các cơ chế và kĩ thuật được sử dụng bởi malware cố gắng ẩn nấp, trốn tránh không bị phát hiện bởi các chương trình chống spyware, virus và các tiện ích hệ thống. Thực ra, rootkit tự bản thân không mang tính hiểm độc nhưng khi chúng được sử dụng cùng với các chương trình mang tính "phá hoại" như: virus, sâu, phần mềm gián điệp, trojan... thì lại nguy hiểm hơn rất nhiều.

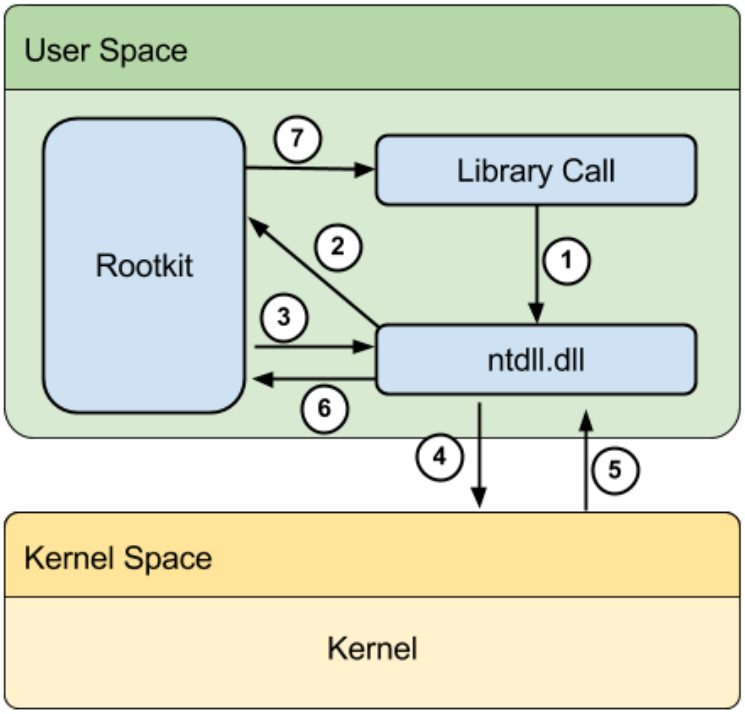
Computer security ring



Cơ chế hoạt động Rootkit

Rootkit được phân loại dựa trên sự duy trì sau khi khởi động lại hoặc hoạt động ở chế độ người dùng (*user mode*) hay ở chế độ cấp hệ thống (*kernel mode*) nên sẽ có cơ chế hoạt động khác nhau.

Rootkit chế độ người dùng (User-mode Rootkits)Rootkit ở chế độ người dùng (***User-mode*** ring 3)sẽ can thiệp tất cả các hàm gọi hệ thống API (Application Programming Interface - Giao tiếp lập trình ứng dụng) như: DLL injection in Windows(FindFirstFile/FindNextFile, ntdll.dll), File Hiding in Linux(find) các rootkit này sẽ chặn các hàm này và thay đổi các kết quả dữ liệu đầu ra nhằm loại bỏ các tập tin chứa rootkit khỏi kết quả trả về mô hình như sau:



Một số dạng user mode-rootkit khác: chiếm quyền điều khiển từ xa(remote access), giả danh chiếm quyền root bằng việc thay đổi các hàm như su, passwd

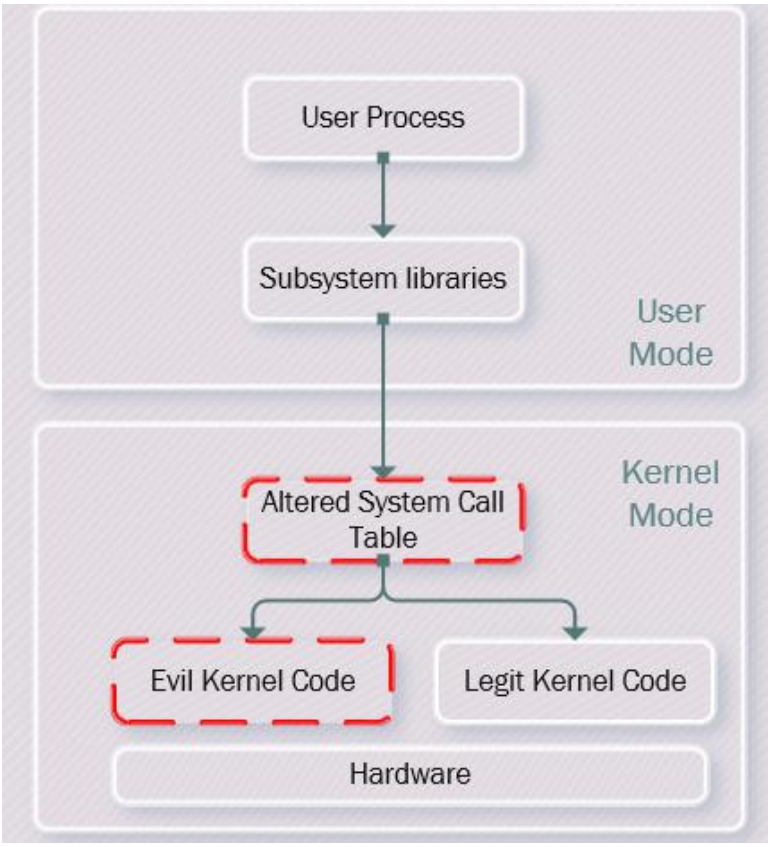
Rootkit chế độ nhân (Kernel-mode Rootkits)

Rootkit chế độ nhân nguy hiểm hơn các loại trên, chúng không chỉ chặn các hàm API hệ thống mà còn có thể thao tác trực tiếp các cấu trúc dữ liệu hệ thống trong chế độ nhân(System Call Table) như sau

The kernel sits between individual applications and underlying Hardware. Kernel mode and user mode processes runs at different level or as they call it rings with ring 0 being the most sensitive level and user mode resides in ring 3, which is the least sensitive level. The applications that reside in ring 3 cannot directly access more secure ring i.e. kernel in ring 0. Instead, they rely on system calls which they execute using subsystem libraries like ntdll.dll, so the flow is like **User Mode -> System Libraries -> System Call Table -> Kernel** where system call table is used to map the code branch for each system call inside kernel.
Now as we have refreshed the concept of Kernel and system calls, let focus back onto the Kernel Mode Rootkit. To implement Kernel Mode rootkit, attacker will alter the kernel.

- Since the System Call Table is used to map the kernel code, what the attacker gets hold of in this system is the call table. The attacker can use insmod to do that, and then map malicious instructions. So the flow would be **User Mode -> System Libraries -> Altered System Call Table.**
- The attacker can then insert malware (methods to be discussed later) and then execute the evil kernel code. So the flow will be like: **User Mode -> System Libraries -> Altered System Call Table > Evil Kernel Code.**

This all looks like below:



Now we need to know how this system call table can be altered and malicious kernel code can be inserted. This can be done in the following ways: external kernel module in Linux, device drivers in Windows

Câu 17: Mô tả một tấn công vào giao thức mã hoá WEP mà anh chị biết

Tấn công này là tấn công lắng nghe, thu thập dữ liệu, nghe lén dữ liệu

WEP sử dụng RC4, một thuật toán sử dụng phương thức mã hóa dòng (stream cipher), nên cần một cơ chế đảm bảo hai dữ liệu giống nhau sẽ không cho kết quả giống nhau sau khi được mã hóa hai lần khác nhau một giá trị có tên Initialization Vector (IV) được sử dụng để cộng thêm với khóa nhằm tạo ra khóa khác nhau mỗi lần mã hóa. IV là một giá trị có chiều dài 24 bit và được chuẩn IEEE 802.11 đề nghị (không bắt buộc) phải thay đổi theo từng gói dữ liệu. Vì máy gửi tạo ra IV không theo định luật hay tiêu chuẩn, IV bắt buộc phải được gửi đến máy nhận ở dạng không mã hóa. Máy nhận sẽ sử dụng giá trị IV và khóa để giải mã gói dữ liệu.

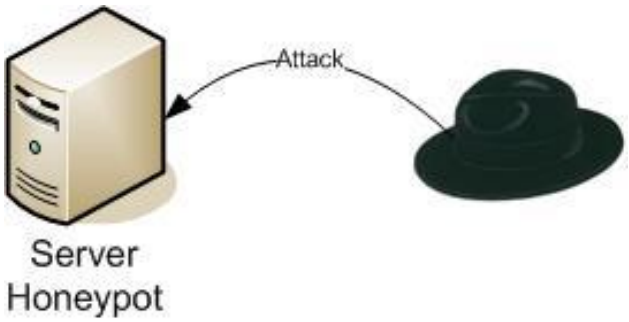
Do giá trị IV được truyền đi ở dạng không mã hóa và đặt trong header của gói dữ liệu 802.11 nên bất cứ ai "tóm được" dữ liệu trên mạng đều có thể thấy được. Với độ dài 24 bit, giá trị của IV dao động trong khoảng 16.777.216 trường hợp hacker có thể bắt gói dữ liệu và tìm ra được khóa WEP. Thêm vào đó, ba nhà phân tích mã hóa Fluhrer, Mantin và Shamir (FMS) đã phát hiện thêm những điểm yếu của thuật toán tạo IV cho RC4. FMS đã vạch ra một phương pháp phát hiện và sử dụng những IV lỗi nhằm tìm ra khóa WEP.

Thêm vào đó, một trong những mối nguy hiểm lớn nhất là những cách tấn công dùng hai phương pháp nêu trên đều mang tính chất thụ động. Có nghĩa là kẻ tấn công chỉ cần thu nhận các gói dữ liệu trên đường truyền mà không cần liên lạc với Access Point. Điều này khiến khả năng phát hiện các tấn công tìm khóa WEP đầy khó khăn và gần như không thể phát hiện được.

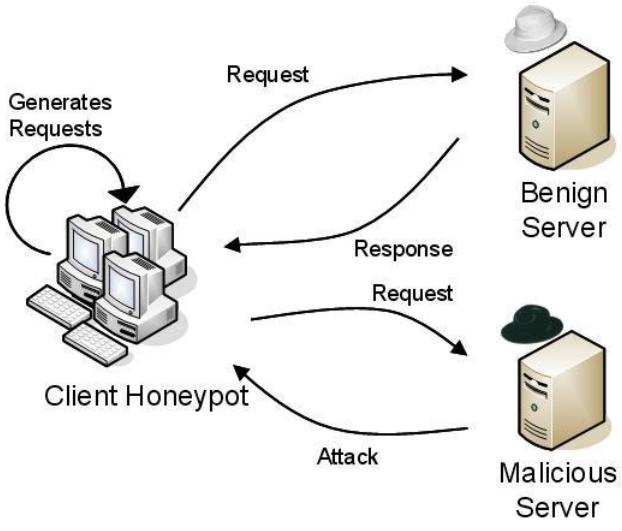
Hiện nay, trên Internet đã sẵn có những công cụ có khả năng tìm khóa WEP như AirCrack AirSnort, dWepCrack, WepAttack, WepCrack, WepLab.

Câu 18. Trình bày tổng quan về client side honeypot và server side honeypot. Giải thích hoạt động của honeypot Nepenthese

Server side honeypot: thuộc kiểu tấn công server side attack tức là máy chủ sẽ cung cấp một số dịch vụ(web, ftp, streaming) để tương tác với người dùng, kẻ tấn công sẽ khai thác các lỗi bảo mật trên máy chủ đó để thực hiện các cuộc tấn công, chính vì lý do đó server side honeypot giả lập một hoặc nhiều dịch vụ tương tự như thật trên máy vật lý hoặc máy ảo để nhận các cuộc tấn công sau đó dữ liệu mạng được thu thập và phân tích, nhận dạng, truy vết ra người tấn công để cảnh báo và bảo vệ hệ thống server thật. Ví dụ: Honeyd, Roo Honeywall



Client side honeypot: thuộc kiểu tấn công client side attack tức là tấn công dựa trên lỗ hổng ứng dụng của người dùng như trình duyệt, ứng dụng gửi mail do thực hiện tương tác(request/response) với một máy chủ nguy hiểm(malicious server), ngược lại với server side honeypot, client side honeypot được cài đặt trên máy người dùng để thực hiện giả lập tương tác với một máy chủ nguy hiểm(malicious server) thông qua những giao thức cơ bản: http, ftp, udp để thu thập và phân tích, nhận dạng, truy vết các cuộc tấn công để cảnh báo và bảo vệ hệ thống server thật, ví dụ: HoneyC or wget



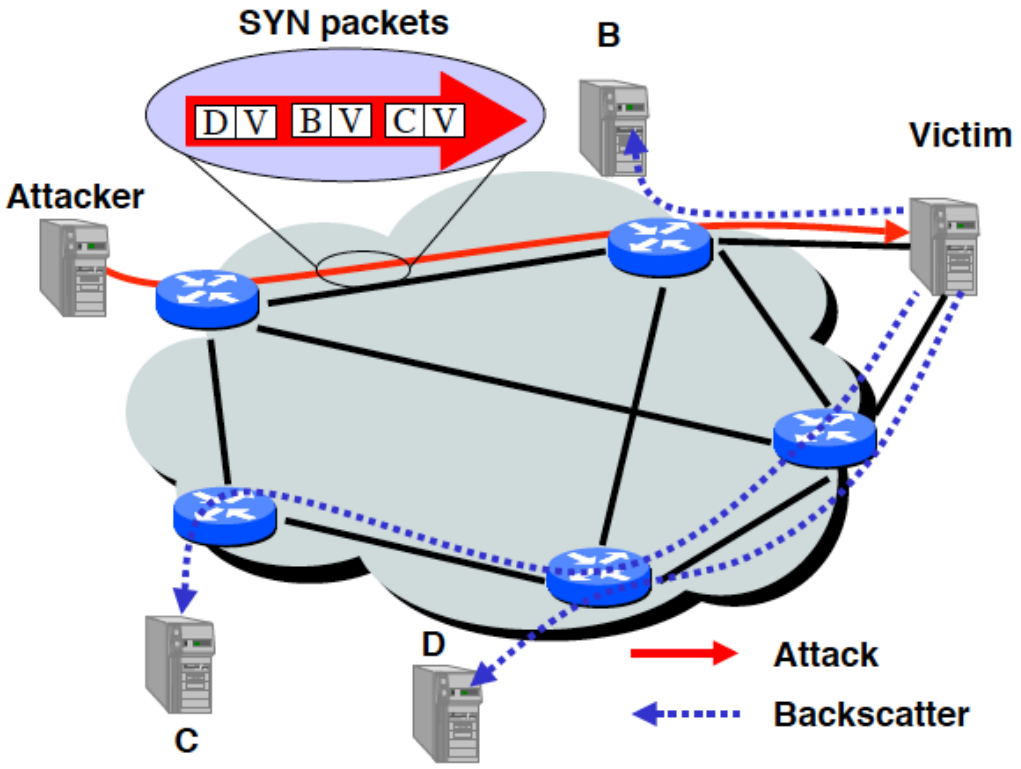
Giải thích hoạt động của honeypot Nepenthese

Nepenthese còn được gọi là công cụ thu thập mã độc(**malware collector)** là một **honeypot** loại tương tác thấp (low interaction) chạy trên server UNIX, cung cấp đủ kiểu dịch vụ mô phỏng để lừa các cuộc tấn công tự động. **Nepenthes** sẽ cố gắng thu thập thật nhiều payload độc hại và cung cấp tuỳ chọn tự động đưa nó tới sandbox ví dụ như Norman sandbox nơi lưu trữ và phân tích rất nhiều tập tin mã độc. Sau đó bạn sẽ nhận được bản ghi thuộc tính malware gửi đến theo địa chỉ e-mail cung cấp(submit-file.conf).

Nếu chạy Nepenthes trên máy mở, bạn sẽ nhanh chóng biết được có bao nhiêu malware trôi nổi trên net. Hiện nay có nhiều biến thể khác nhau của một vài gia đình bot như: SpyBot, Agobot… Nhiều chương trình antivirus vẫn chưa dò tìm được một số lượng đáng kể các bot này. Có thể mọi người không mấy vui vẻ, nhưng Nepenthes lại trở nên đặc biệt hữu ích.

Câu 19: Trình bày cách thức xác định mức độ tấn công từ chối dịch vụ dựa vào gói tin backscatter. Phân tích các ưu và nhược điểm của hướng tiếp cận xác định tấn công từ chối dịch vụ được mô tả trong công trình:"Inferring Internet Denial-of-Service Activity"

(Mô tả lại khái niệm mức độ tấn công của chị Hiền) -- Kiên
3.1 Backscatter analysis (Tài liệu Inferring_Internet_DoS_Activity.pdf của thầy)
Phân tích các gói tin Backscatter.
Giả sử địa chỉ nguồn (địa chỉ giả mạo của kẻ tấn công) là ngẫu nhiên cho các gói tin trong một cuộc tấn công. Khi hacker gửi m gói tin thì xác suất 1 host trong mạng (B, C or D) nhận được các gói tin không mong muốn từ máy nạn nhân trả về là m/2^32 gói tin. Tương tự nếu như ta giám sát trên n địa chỉ IP thì ta sẽ thấy được E(X)=n*m/2^32
Bằng một cách nào đó nếu ta theo dõi được một lượng đủ lớn các địa chỉ IP chúng ta sẽ thu lượm được các mẫu về denial-of-service activity. Những mẫu này chính là thông tin về máy nạn nhân, thông tin về loại hình tấn công, và một timestamp từ đó có thể ước tính được thời gian tấn công
Xa hơn nữa, là chúng ta có thể sử dụng tỉ lệ xuất hiện trung bình của các gói tin không mong muốn trên phạm vi các địa chỉ IP mà chúng ta theo dõi để ước tính tỉ lệ thực tế các cuộc tấn công đến máy nạn nhân là: R>=R'*(2^32)/n. (R' là đo lường giá trị trung bình các gói backscatter từ máy nạn nhân, R là giá trị suy đoán được tỉ lệ tấn công dựa trên các gói tin mỗi giây.)



ƯU và nhược điểm (chưa hoàn chỉnh, mn đóng góp thêm)
ƯU điểm:

Nhược điểm của BackScatter:
1/ Dựa vào địa chỉ nguồn
Ngày này các ISPs đã tăng cường lọc địa chỉ nguồn trên các router. Nếu gói tin đi vào có địa chỉ nguồn không thuộc ISP thì nó sẽ drop gói tín đó như vậy nó sẽ làm cho việc theo dõi không chính xác
2/ Trường hợp hacker sử dụng phương thức tấn công phản xạ (gửi gói tin có địa chỉ nguồn là địa chỉ của victim đến bên thứ 3) lúc này phạm vi các máy của bên thứ 3 rất lớn dẫn đến khó xác định chính xác mức độ tấn công
3/ Nếu hacker sử dụng địa chỉ thực thì ta sẽ khó có thể xác định chính xác mức độ tấn công
4/ Trường hợp số lượng lớn lượng gói tin được gửi đến nạn nhân, khi đạt đến 1 ngưỡng nhất định thì các gói tin còn lại sẽ bị drop dẫn đến việc đánh giá bằng backscatter không còn chính xác

Backscatter[edit]

Trong an ninh mạng máy tính, tán xạ (**Backscatter**) là một hiệu ứng của một tấn công từ chối

dịch vụ giả mạo. Trong loại này tấn công, lừa đảo kẻ tấn công (hoặc Forges) địa chỉ nguồn trong gói tin IP gửi đến các nạn nhân. Nói chung, các máy tính nạn nhân không thể phân biệt giữa các gói tin giả mạo và các gói tin hợp pháp, do đó, các nạn nhân phản ứng với các gói tin giả mạo như bình thường. Những gói tin trả lời được gọi là tán xạ. [64]

Nếu kẻ tấn công giả mạo địa chỉ nguồn ngẫu nhiên, các gói tin phản hồi tán xạ từ các nạn nhân sẽ được đưa trở lại khu ngẫu nhiên. Hiệu ứng này có thể được sử dụng bởi các kính thiên văn hệ thống làm bằng chứng gián tiếp của các cuộc tấn công như vậy.

Thuật ngữ "phân tích tán xạ ngược" đề cập đến quan sát các gói tin tán xạ đi đến một phần ý nghĩa thống kê của không gian địa chỉ IP để xác định đặc điểm của các cuộc tấn công DoS và các nạn nhân.

"phân tích tán xạ ngược," (backscatter analysis) để ước lượng các hoạt động tấn công từ chối dịch vụ trên mạng Internet. Sử dụng kỹ thuật này, chúng tôi đã quan sát thấy các cuộc tấn công DoS phổ biến trên mạng Internet, phân bố trong nhiều tên miền và các ISP khác nhau. Các kích thước và độ dài của các cuộc tấn công chúng ta quan sát là heavy tailed , với một số lượng nhỏ của các cuộc tấn công chiếm một phần đáng kể của khối lượng cuộc tấn công tổng thể. Hơn nữa, chúng ta thấy một số lượng đáng ngạc nhiên của các cuộc tấn công nhắm vào một số ít quốc gia nước ngoài, tại máy nhà, và hướng tới dịch vụ Internet đặc biệt

Bằng cách quan sát một với phạm vi địa chỉ đủ lớn chúng ta có thể có được tất cả "mẫu" như tấn công từ chối dịch vụ hoạt động trên Internet. Chứa trong các mẫu là danh tính của các nạn nhân, thông tin về các loại tấn công, và thời gian từ đó chúng ta có thể ước tính thời gian tấn công. Hơn nữa, với những giả định, chúng ta cũng có thể sử dụng tỷ lệ xuất hiện trung bình của phản ứng không mong muốn hướng vào dải địa chỉ theo dõi để đánh giá tỷ lệ thực tế của cuộc tấn công được hướng vào các nạn nhân.

Câu 20: Giải thích tác hại của các thuật toán Pseudorandom Generators không ngẫu nhiên nếu sử dụng tạo keystream trong stream cipher

Theo mình hiểu các giải thuật PRG không ngẫu nhiên là các giải thuật weak PRG (slide 24 – Stream-annotated). Tác hại của nó là hacker có thể đoán được private key, lý giải bằng tiếng anh như bên dưới.

When you generate a private key, you do so with a source of randomness. If that source of randomness can output N different streams of bits, then, at most, you may get N different private key. This is where we like to talk of entropy, which is a measure of how big that N is. When the source of randomness is said to offer "100 bits of entropy", then it means that (roughly) N = 2100.

The attacker will want to obtain your private key. If he knows that you use a weak source with a low N (say, only 40 bits of entropy), then he can, on his own machine, enumerate all possible outputs from the random source and work out the corresponding private key.

For instance, suppose that you used a PRNG seeded with the current time, expressed in microseconds. This is the time as known by your machine. The attacker assumes that your machine is reasonably well set with the current time, say within 10 seconds. So, from the point of view of the attacker, the seed for your PRNG is known within a 10 seconds range; since the PRNG use the time in microseconds, that leaves him with N = 10000000 possible seeds. The attacker then says to himself: "IF that guy used as seed value x, THEN his code produced private key value Kx; let's see if that matches his public key... nope. So he did not use x. Let's try again with x+1 (and so on)."

So a weak PRNG is deadly in such situations.

Nguồn: http://security.stackexchange.com/questions/42327/how-does-a-weakness-in-a-random-number-generator-lead-to-a-compromise-of-the-ent

Or Slide 15 / 02-stream-v2-annotated.pptx – Thầy Hậu