BÁO CÁO THỰC HÀNH

**Môn học: Thực hành Cơ chế hoạt động của mã độc**

**Lab 5: Kernel Rootkits**

*GVHD: Nguyễn Hữu Quyền*

1. **THÔNG TIN CHUNG:**

*(Liệt kê tất cả các thành viên trong nhóm)*

Lớp: NT230.N21.ATCL.1

Nhóm: 3

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** | **Email** |
| 1 | Trần Tấn Tài | 20521862 | 20521862@gm.uit.edu.vn |
| 2 | Bạch Văn Xuân Thông | 20521978 | 20521978@gm.uit.edu.vn |
| 3 | Phạm Bá Tín | 20522016 | 20522016@gm.uit.edu.vn |

1. **NỘI DUNG THỰC HIỆN:[[1]](#footnote-0)**

|  |  |  |
| --- | --- | --- |
| **STT** | **Công việc** | **Kết quả tự đánh giá** |
| 1 | Yêu cầu 1 | 100% |
| 2 | Yêu cầu 2 | 100% |
| 3 | Yêu cầu 3 | 100% |
| 4 | Yêu cầu 4 | 100% |
| 5 | Yêu cầu 5 | 100% |
| 6 | Yêu cầu 6 | 100% |

**Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.**

BÁO CÁO CHI TIẾT

## Yêu cầu 1. Đọc, giải thích code và xác định entry point của kernel mô-đun (b4rn\_init()) được gọi sau khi mô-đun được load bởi kernel (ví dụ bằng insmod hoặc modprobe).

## Yêu cầu 2. Trả lời câu hỏi sau: Kẻ tấn công có thể sử dụng backdoor do rootkit tiết lộ để truy cập từ xa được không? Giải thích lý do tại sao và tại sao không.

Kẻ tấn công có thể sử dụng backdoor mà rootkit tiết lộ để truy cập từ xa bởi vì:

* Rootkit có khả cung cấp quyền root nên khi kẻ tấn công sử dụng backdoor do rootkit tạo sẽ có thể mở thực hiện được TCP port, qua đó thực hiện bind shell hoặc reverse shell để truy cập từ xa mà không cần phải có sự đồng ý hoặc nhận thức của người dùng
* Backdoor do rootkit tạo có thể được thiết lập như một cổng truy cập từ xa, cho phép kẻ tấn công kết nối và tương tác với hệ thống. Điều này giúp kẻ tấn công cài đặt các phần mềm độc hại khác hoặc truy cập một số dữ liệu.
* Rootkit thường có khả năng ản danh và xóa dấu vết hoạt động độc hại trên hệ thống, do đó gây khó khăn cho người dùng để phát hiện ra các hành động độc hại của kẻ tấn công

## Yêu cầu 3. Nhận thấy rằng đây là một backdoor khá thô sơ. Chắc chắn có nhiều cách khác lén lút hơn để thực hiện việc này (ví dụ: để chúng ta không tạo tập tin device không mong muốn trên hệ thống). Hãy tìm cách khác để thực hiện điều đó.

- Để chúng ta không tạo tập tin device không mong muốn trên hệ thống, chúng ta sẽ thực hiện hook một system call như lệnh system call ‘open()’. Điều này có thể được thực hiện bằng cách sửa đổi bảng system call, chứa địa chỉ của tất cả các system call trong kernel

## Yêu cầu 4. (BTVN) Giải thích lý do tại sao chúng ta phải:

## (1) sử dụng con trỏ hàm (function pointer) và hàm kallsyms\_\*() để gọi một số thường trình nhất định (certain routines)

## (2) thao tác cr0 và bảo vệ trang (page protections) để cài đặt phần ghi đè hàm (function overrides) của chúng ta.

* **(1):**
* Thường trình nhất định (certain routines) là các chức năng hoặc tính năng được cài đặt sẵn trên hệ thống máy tính và thường được sử dụng trong quá trình vận hành và quản lý các thành phần khác nhau của hệ thống, bao gồm các tác vụ quản lý bộ nhớ, xử lý tệp, định tuyến mạng và nhiều tính năng khác. Chúng thường được cấu hình và gọi khi hệ thống khởi động hoặc khi có các tác vụ và yêu cầu được thực hiện trên hệ thống.
* Các thường trình nhất định trong chương trình *b4rnd00r.c* bao gồm hàm *sys\_getuid()* và hàm *sys\_kill()* thông qua syscall\_table. Đây là các hàm đặc biệt được cài đặt sẵn trong hệ thống Linux và được sử dụng để lấy thông tin về ID người dùng và gửi yêu cầu giết tiến trình trên hệ thống.
* Tuy nhiên, việc gọi các hàm này trực tiếp và không che giấu có thể dẫn đến việc chúng ta để lại dấu vết trên hệ thống, từ đó dễ dàng bị phát hiện. Vì vậy, chúng ta cần sử dụng con trỏ hàm và hàm kallsyms\_\*() để giấu các cuộc gọi của chúng ta và thực hiện các hoạt động một cách an toàn và bảo mật hơn.
* Việc sử dụng con trỏ hàm giúp chúng ta gọi các hàm mà không cần phải khai báo và tải trực tiếp chúng vào hệ thống.
* Trong khi đó, việc sử dụng hàm kallsyms\_\*() cho phép chúng ta tìm kiếm và truy cập vào các địa chỉ của các hàm và ký hiệu trên kernel của hệ thống.
* Tóm lại, việc sử dụng con trỏ hàm và hàm kallsyms\_\*() là một cách để giấu các hoạt động xâm nhập và kiểm soát khỏi các phát hiện của hệ thống . Việc này cho phép cho chúng ta có thể truy cập vào các chức năng hệ thống và các tập tin của nó một cách bí mật, trong khi vẫn giữ được tính năng và khả năng hoạt động của hệ thống.
* **(2):**
* Khi muốn ghi đè (Override) một hàm đã có sẵn trong kernel thì ta cần phải truy cập trực tiếp vào bộ nhớ của kernel để thay đổi mã của nó. Tuy nhiên, bộ nhớ của kernel là một khu vực được bảo vệ chặt chẽ để đảm bảo an toàn và bảo mật của hệ thống. Do đó, để có thể ghi đè hàm trong kernel, chúng ta cần phải thực hiện thay đổi giá trị của thanh ghi control register 0 (CR0) và bảo vệ trang (page protections) của kernel để can thiệp vào các cơ chế bảo vệ này.
* Thay đổi giá trị của thanh ghi CR0: Thanh ghi CR0 được sử dụng trong kiến trúc x86 để kiểm soát các tính năng bảo mật của hệ thống, bao gồm việc bật/tắt các cơ chế bảo vệ trang (page protections).
* Bảo vệ trang (page protections): Kernel sử dụng các trang bộ nhớ để quản lý các vùng nhớ khác nhau và bảo vệ chúng khỏi truy cập trái phép. Vì vậy, đối với các vùng nhớ của kernel được bảo vệ, việc ghi đè trực tiếp vào chúng là không thể. Để giải quyết vấn đề này, chương trình b4rnd00r.c sử dụng kỹ thuật bảo vệ trang để tạm thời vô hiệu hóa bảo vệ trang trong các trang bộ nhớ mà nó muốn thay đổi.
* Cụ thể trong chương trình *b4rnd00r.c:*
* Hàm unprotect\_page() được sử dụng để vô hiệu hóa bảo vệ trang của một trang bộ nhớ tại vị trí địa chỉ addr. Hàm sẽ tắt cờ Write Protect (WP) của thanh ghi CR0.
* Một khi giá trị của thanh ghi CR0 đã được thay đổi và các trang bộ nhớ đã được bảo vệ tạm thời, chương trình b4rnd00r.c có thể ghi đè vào mã của các hàm trong kernel để thực hiện các thao tác như muốn.
* Hàm protect\_page() lại được sử dụng để kích hoạt lại bảo vệ trang cho một trang bộ nhớ tại địa chỉ addr. Hàm sẽ bật lại cờ Write Protect (WP) của thanh ghi CR0.
* Khi các thay đổi đã được hoàn tất, chương trình sẽ phục hồi giá trị ban đầu của thanh ghi CR0 và bảo vệ trang của kernel để trả lại tình trạng bảo mật ban đầu của hệ thống. Việc này giúp đảm bảo sự ổn định và an toàn của hệ thống khi thực hiện các thao tác ghi đè hàm.

## Yêu cầu 5. (BTVN) Giả sử ta muốn gây khó khăn cho quản trị viên hệ thống trong việc xóa rootkit của ta ra khỏi hệ thống. Vậy có thể làm gì để ngăn chặn điều đó? (Gợi ý: có lệnh gọi hệ thống reboot()).

* Một cách để ngăn chặn quản trị viên hệ thống xóa rootkit của ta là khiến hệ thống tự động khởi động lại sau khi quản trị viên cố gắng xóa rootkit.

#include <linux/reboot.h>

#include <sys/reboot.h>

// Call reboot system call

reboot(LINUX\_REBOOT\_CMD\_RESTART);

* Thư viện *linux/reboot.h* chứa các hằng số (**`***#define*) và khai báo (declaration) cho các syscall liên quan đến việc khởi động lại hệ thống. Trong đó hằng số LINUX\_REBOOT\_CMD\_RESTART dùng để khởi động lại hệ thống.
* Syscall rebootđược thực hiện sau khi có đặc quyền root.

## Yêu cầu 6. (BTVN): Thực hiện lại các bước trên và đưa ra kết quả, giải thích.

1. Ghi nội dung công việc, các kịch bản trong bài Thực hành [↑](#footnote-ref-0)