УДК 004.4
DOI https://doi.org/10.32689/maup.it.2024.2.9

Дмитро ОЛЬХОВСЬКИЙ
кандидат фізико-математичних наук,
доцент кафедри комп'ютерних наук та інформаційних технологій,
Полтавський університет економіки і торгівлі, dmitriy@olhovsky.name
***ORCID**: 0000-0003-0313-6977*

Давід ЛИСЕНКО
здобувач освіти напрямку «Комп'ютерні науки»,
Полтавський університет економіки і торгівлі, david.lysenko95@gmail.com
***ORCID:** 0009-0008-7914-0343*

Андрій ЖУЛЯ
аспірант,
Полтавський університет економіки і торгівлі, andreyzhulya@gmail.com
***ORCID:** 0009-0007-5112-0490*

АНАЛІЗ БЕЗПЕКИ ТА МЕТОДИ ЗАХИСТУ ХМАРНОЇ ІНФРАСТРУКТУРИ НА ПРИКЛАДІ ROOTKIT ДЛЯ ЯДРА ОПЕРАЦІЙНОЇ СИСТЕМИ

Анотація. *Стаття висвітлює основні аспекти розробки та аналізу rootkit для операційної системи Linux, зокрема, розглядаються методи і технології, які використовуються для створення rootkit-ів, а також виклики, що стоять перед безпекою операційних систем. У контексті зростаючої складності та критичності інформаційної безпеки, робота підкреслює необхідність розуміння та виявлення rootkit, що дозволить підвищити захист систем від потенційних загроз.*

Мета статті *полягає в детальному розгляді процесу розробки rootkit для операційної системи Linux, аналізі його функціональних можливостей, оцінці впливу на безпеку системи та розробці рекомендацій щодо захисту від подібних загроз. Важливим аспектом є дослідження методів приховування процесів, файлів, мережевих з'єднань та інших об'єктів у системі.*

Методологія *включає розробку rootkit на мові програмування C з використанням модулів ядра Linux та командного інтерфейсу користувача (CLI). Основні інструменти, використані під час розробки, включають GCC (GNU Compiler Collection), який є стандартним компілятором для мови програмування C у середовищі Linux, GDB (GNU Debugger) для налагодження коду, Makefile для автоматизації процесу компіляції та збірки модулів ядра, а також Kernel Headers та Kernel Source, які необхідні для розробки модулів ядра.*

Наукова новизна*. У статті представлено глибокий аналіз та практичну реалізацію rootkit для операційної системи Linux, що є вагомим внеском у галузь інформаційної безпеки. Вперше детально розглянуто процес проектування та розробки rootkit, включаючи аналіз функціональних вимог, проектування модулів ядра, приховування процесів та файлів, а також розробку механізмів для отримання суперкористувацьких привілеїв. Значна увага приділена методам оптимізації rootkit-у для мінімізації його впливу на продуктивність системи, а також аналізу способів його виявлення та нейтралізації. Це дослідження надає нові знання про методи створення та приховування rootkit-ів, що допомагає у розробці більш ефективних засобів захисту інформаційних систем.*

Висновки*. У ході виконання дослідження було досягнуто важливих результатів, які мають практичне значення для підвищення рівня безпеки операційних систем Linux. В результаті роботи було розроблено rootkit, який демонструє можливості приховування процесів та файлів, отримання суперкористувацьких привілеїв та організацію зворотного shell. Проведено детальний аналіз впливу rootkit на безпеку системи та розроблено рекомендації щодо його виявлення та нейтралізації. Запропоновані методи виявлення та захисту від rootkit-ів сприяють підвищенню рівня інформаційної безпеки та можуть бути використані в практичній діяльності з захисту комп'ютерних систем.*

Ключові слова: *Linux, безпека системи, приховування процесів, отримання привілеїв, зворотний shell, інформаційна безпека.*

Dmytro OLHOVSKY, David LYSENKO, Andrey ZHULYA. SECURITY ANALYSIS AND CLOUD INFRASTRUCTURE PROTECTION METHODS USING ROOTKIT FOR OPERATING SYSTEM KERNEL

Abstract. *This article highlights the key aspects of developing and analyzing a rootkit for the Linux operating system, particularly focusing on the methods and technologies used to create rootkits, as well as the challenges posed to operating system security. In the context of increasing complexity and criticality of information security, this article emphasizes the necessity of understanding and detecting rootkits to enhance system protection against potential threats.*

The aim of the article *is to provide a detailed examination of the process of developing a rootkit for the Linux operating system, analyzing its functional capabilities, assessing its impact on system security, and developing recommendations for protection against such threats. An important aspect is the study of methods for hiding processes, files, network connections, and other objects in the system.*

The research methodology *includes the development of a rootkit in the C programming language using Linux kernel modules and a command-line interface (CLI). The main tools used during development include GCC (GNU Compiler Collection), the standard compiler for the C programming language in the Linux environment; GDB (GNU Debugger) for code debugging;*

Makefile for automating the compilation and assembly process of kernel modules; and Kernel Headers and Kernel Source, which are necessary for kernel module development.

Scientific novelty. *This article presents a comprehensive analysis and practical implementation of a rootkit for the Linux operating system, which is a significant contribution to the field of information security. For the first time, the process of designing and developing a rootkit is examined in detail, including the analysis of functional requirements, kernel module design, process and file hiding, and the development of mechanisms for obtaining superuser privileges. Considerable attention is given to methods of optimizing the rootkit to minimize its impact on system performance, as well as to analyzing ways of detecting and neutralizing it. This research provides new insights into the methods of creating and hiding rootkits, aiding in the development of more effective means of protecting information systems.*

Conclusions. *During the course of this research, significant results were achieved that have practical implications for enhancing the security of Linux operating systems. As a result of the work, a rootkit was developed that demonstrates capabilities for hiding processes and files, obtaining superuser privileges, and organizing a reverse shell. A detailed analysis of the rootkit's impact on system security was conducted, and recommendations for its detection and neutralization were developed. The proposed methods for detecting and protecting against rootkits contribute to improving the level of information security and can be used in practical activities for protecting computer systems.*

Key words: *Linux, system security, process hiding, privilege escalation, reverse shell, information security.*

Вступ. Постановка проблеми в загальному вигляді та її зв'язок з важливими науковими чи практичними завданнями. Rootkit-и представляють собою одну з найбільш небезпечних категорій шкідливого програмного забезпечення, яке здатне приховувати свою присутність у системі, забезпечуючи зловмиснику постійний доступ до компрометованого пристрою. Вони можуть модифікувати системні процеси, приховувати файли та мережеві з'єднання, а також перехоплювати дані, що передаються по мережі. У зв'язку з цим, проблема виявлення та нейтралізації rootkit-ів є критично важливою для забезпечення інформаційної безпеки операційних систем, зокрема Linux, яка широко використовується у різних сферах, включаючи серверні середовища, наукові обчислення та інфраструктуру хмарних обчислень.

Останнім часом зростає кількість атак, що використовують rootkit-и, для досягнення своїх цілей, що ускладнює завдання забезпечення належного рівня безпеки інформаційних систем. Існуючі методи захисту часто виявляються недостатньо ефективними у боротьбі з сучасними rootkit-ами, які постійно вдосконалюються і використовують нові техніки приховування.

Основною проблемою є те, що rootkit-и можуть впроваджуватися на рівні ядра операційної системи, що робить їх важкодоступними для традиційних антивірусних програм і систем виявлення вторгнень. Це обумовлює необхідність розробки нових методів виявлення, аналізу та нейтралізації rootkit-ів, що можуть забезпечити більш високий рівень захисту інформаційних систем.

Таким чином, необхідно провести детальний аналіз існуючих методів розробки та виявлення rootkit-ів, дослідити їх вплив на безпеку операційних систем та розробити нові підходи до забезпечення захисту від цього типу шкідливого програмного забезпечення. Важливим аспектом є також розробка рекомендацій для системних адміністраторів та фахівців з інформаційної безпеки щодо ефективних методів виявлення та нейтралізації rootkit.

Аналіз останніх досліджень і публікацій. У сфері інформаційної безпеки проведено значну кількість досліджень, присвячених проблемам виявлення та нейтралізації rootkit-ів [1-10]. Наукові роботи та публікації останніх років підкреслюють важливість дослідження технік приховування, які використовуються rootkit-ами, а також розробки нових методів для їх виявлення та запобігання.

Одним з ключових напрямків досліджень є аналіз методів приховування, що використовуються rootkit-ами на рівні ядра операційної системи Linux. В [7], [9] розглядаються різні техніки модифікації системних викликів та маніпуляції з даними ядра, що дозволяють rootkit-ам залишатися непоміченими стандартними засобами захисту. Такі техніки включають перехоплення системних викликів (syscall hooking), маніпуляцію з таблицею системних викликів (System Call Table) та використання прихованих модулів ядра (kernel modules).

Іншим важливим аспектом досліджень є розробка ефективних методів виявлення rootkit-ів. У працях [3], [4] запропоновано використання різних методів аналізу поведінки системи та аномалій для виявлення прихованих процесів та файлів. Зокрема, розглядаються методи динамічного аналізу, які базуються на відстеженні поведінки програм у режимі реального часу, а також методи статичного аналізу, що передбачають аналіз коду та структури файлів.

У дослідженнях [2], [5] також акцентується увага на використанні машинного навчання та штучного інтелекту для виявлення rootkit-ів. Використання алгоритмів машинного навчання дозволяє значно підвищити ефективність виявлення за рахунок автоматизованого аналізу великих обсягів даних та виявлення патернів, що свідчать про наявність rootkit-у.

Крім того, в роботах [1], [10] розглядаються методи нейтралізації rootkit-ів, включаючи розробку спеціалізованого програмного забезпечення для очищення системи від шкідливого коду та відновлення її

нормальної роботи. Значна увага приділяється також питанням забезпечення безпеки під час розробки та експлуатації операційних систем, що включає використання захищених компіляторів, перевірки коду та інших заходів безпеки.

Таким чином, аналіз останніх досліджень та публікацій свідчить про те, що проблема виявлення та нейтралізації rootkit-ів є актуальною та потребує комплексного підходу, який включає дослідження технік приховування, розробку ефективних методів виявлення та нейтралізації, а також впровадження заходів безпеки на всіх етапах життєвого циклу інформаційної системи.

Постановка завдання. Мета статті полягає в детальному аналізі процесу розробки та функціональних можливостей rootkit-у для операційної системи Linux, а також оцінці його впливу на безпеку системи.

Виклад основного матеріалу дослідження. У дослідженні розглядаються основні аспекти розробки та функціонування rootkit-у для операційної системи Linux.

Розробка rootkit-у здійснювалася на мові програмування C з використанням модулів ядра Linux та командного інтерфейсу користувача (CLI). Основними інструментами, використаними під час розробки, були:

– GCC (GNU Compiler Collection) – стандартний компілятор для мови програмування C у середовищі Linux.

– GDB (GNU Debugger) – для налагодження коду.

– Makefile – для автоматизації процесу компіляції та збірки модулів ядра.

– Kernel Headers та Kernel Source – необхідні для розробки модулів ядра.

Процес розробки включав наступні етапи:

1. Створення модулів ядра для Linux:

– Було створено та оптимізовано модулі ядра для забезпечення основної функціональності rootkit-у.

– Модулі ядра дозволяють здійснювати приховування процесів та файлів, перехоплення системних викликів та маніпуляції з ними.

```
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>

static int __init lkm_example_init(void) {
    printk(KERN_INFO "Loading LKM Example Module...\n");
    return 0;
}

static void __exit lkm_example_exit(void) {
    printk(KERN_INFO "Unloading LKM Example Module...\n");
}

module_init(lkm_example_init);
module_exit(lkm_example_exit);

MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("LKM Example Module");
MODULE_AUTHOR("Author Name");
```

2. Реалізація приховування процесів та файлів:

– Rootkit здатен приховувати певні процеси від користувачів та системних інструментів моніторингу, що робить їх невидимими для адміністратора системи.

– Файли та директорії можуть бути приховані за допомогою маніпуляції зі структурою файлової системи.

```
struct linux_dirent {
    unsigned long d_ino;
    unsigned long d_off;
    unsigned short d_reclen;
    char d_name[];
};

asmlinkage int (*original_getdents)(unsigned int, struct linux_dirent *, unsigned int);
```

```
asmlinkage int hacked_getdents(unsigned int fd, struct linux_dirent *dirp, unsigned int count) {
  int nread = original_getdents(fd, dirp, count);
  if (nread == -1) return -1;

  struct linux_dirent *d;
  int bpos;
  for (bpos = 0; bpos < nread;) {
    d = (struct linux_dirent *) ((char *) dirp + bpos);
    if (strstr(d->d_name, "hidden_file")) {
      memmove(d, (char *) d + d->d_reclen, nread - bpos - d->d_reclen);
      nread -= d->d_reclen;
    } else {
      bpos += d->d_reclen;
    }
  }
  return nread;
}
```

3. Реалізація зворотного shell та його автоматизація:

– Rootkit надає можливість створення зворотного shell, що дозволяє віддаленому зловмиснику отримувати доступ до системи.

– Процес зворотного shell автоматизований, що підвищує його ефективність та зручність використання.

```
#define IP_ADDRESS "192.168.1.1"
#define PORT 4444

void reverse_shell(void) {
  struct sockaddr_in sa;
  int s = socket(AF_INET, SOCK_STREAM, 0);
  sa.sin_family = AF_INET;
  sa.sin_addr.s_addr = inet_addr(IP_ADDRESS);
  sa.sin_port = htons(PORT);
  connect(s, (struct sockaddr *)&sa, sizeof(sa));
  dup2(s, 0);
  dup2(s, 1);
  dup2(s, 2);
  execl("/bin/sh", "sh", NULL);
}
```

4. Оптимізація роботи rootkit-у:

– Було проведено ряд заходів з оптимізації rootkit-у для мінімізації його впливу на продуктивність системи.

– Оптимізація включала зниження обсягу використовуваної пам'яті та CPU, а також підвищення стабільності роботи rootkit-у.

5. Відладка та виправлення помилок:

– У процесі розробки проводилася ретельна відладка коду та виправлення виявлених помилок для забезпечення стабільної роботи rootkit-у.

– Використовувалися методи статичного та динамічного аналізу коду для виявлення потенційних вразливостей.

Нижче наведена блок-схема роботи rootkit-у, що включає процеси приховування файлів, приховування процесів, та реалізації зворотного shell (рис. 1).

Розроблений rootkit успішно демонструє можливості приховування процесів та файлів, отримання суперкористувацьких привілеїв та організацію зворотного shell. Проведений аналіз впливу rootkit-у на безпеку системи показав, що такий rootkit може серйозно загрожувати цілісності та конфіденційності даних в операційній системі Linux.

Було розроблено рекомендації щодо захисту від подібних загроз, включаючи методи виявлення та нейтралізації rootkit-ів. Зокрема, запропоновано використовувати інструменти для моніторингу системних викликів, перевірки цілісності файлів та активного аналізу поведінки системи.

Запропоновані методи виявлення та захисту від rootkit-ів сприяють підвищенню рівня інформаційної безпеки та можуть бути використані в практичній діяльності захисту комп'ютерних систем.

Висновки з даного дослідження та перспективи подальших розвідок у даному напрямі. Таким чином, створено ефективний rootkit для операційної системи Linux, який здатний приховувати процеси та файли, отримувати суперкористувацькі привілеї та організовувати зворотний shell. Реалізовано ефективні методи приховування процесів та файлів, що робить їх невидимими для стандартних системних інструментів моніторингу. Зворотний shell було автоматизовано, що значно спрощує його використання та підвищує ефективність отримання віддаленого доступу до системи з правами суперкористувача. Проведено оптимізацію rootkit-у для мінімізації його впливу на продуктивність системи, включаючи зниження використання ресурсів CPU та пам'яті. Розроблено рекомендації щодо виявлення та нейтралізації rootkit-у, включаючи використання інструментів для моніторингу системних викликів та перевірки цілісності файлів. Запропоновані методи та результати дослідження сприяють підвищенню рівня захисту операційних систем Linux від потенційних загроз. У подальшому планується удосконалення rootkit-у шляхом впровадження додаткових технік для персістенсу та приховування файлів. Це включатиме розширення можливостей rootkit-у для автоматичного встановлення з подальшою чисткою логів, що забезпечить ще більшу непомітність його дій у системі. Також передбачається дослідження та впровадження нових методів приховування мережевої активності, що дозволить зловмиснику залишатися непоміченим під час взаємодії з компрометованою системою.

Окрім того, планується інтеграція механізмів самозахисту rootkit-у від виявлення та нейтралізації, що включатиме динамічну зміну сигнатур та обфускацію коду. Це ускладнить виявлення rootkit-у за допомогою стандартних антивірусних програм та систем виявлення вторгнень.

Рис. 1. Блок-схема роботи rootkit

Список використаних джерел:

1. Barak A. Linux Kernel Programming, Part 2 - Char Device Drivers and Kernel Synchronization. Packt Publishing, 2021. 458 p.
2. Blunden B. Rootkit Arsenal: Escape and Evasion in the Dark Corners of the System. Jones & Bartlett Learning, 2013. 784 p.
3. Bovet D. P., Cesati M. Understanding the Linux Kernel (3rd ed.). O'Reilly Media, 2005. 944 p.
4. Corbet J., Rubini A., Kroah-Hartman G. Linux Device Drivers (3rd ed.). O'Reilly Media, 2005. 640 p.
5. Hoglund G., Butler J. Rootkits: Subverting the Windows Kernel. Addison-Wesley Professional, 2005. 512 p.
6. Ionescu A. Rootkit Uncovered. Security Research Group, 2017. 504 p.
7. Love R. Linux Kernel Development (3rd ed.). Addison-Wesley Professional, 2010. 464 p.
8. Maxwell D. Hacking: The Art of Exploitation. No Starch Press, 2016. 488 p.
9. Raj A., Patel D. Programming the Linux Kernel. Packt Publishing, 2018. 368 p.
10. Vasileios M., Xenofon S. Mastering Linux Security and Hardening. Packt Publishing, 2018. 372 p.