МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО

МЕТОДИЧНІ ВКАЗІВКИ

ЩОДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ

З НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

«ІНФОРМАЦІЙНА СТІЙКІСТЬ КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ ТА МЕРЕЖ»

ДЛЯ СТУДЕНТІВ ДЕННОЇ ФОРМИ НАВЧАННЯ

ЗІ СПЕЦІАЛЬНОСТІ

123 – «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ»

ОСВІТНЬОГО СТУПЕНЯ «МАГІСТР»

ЧАСТИНА І

КРЕМЕНЧУК 2018

Методичні вказівки щодо виконання лабораторних робіт з навчальної дисципліни «Інформаційна стійкість комп'ютерних технологій та мереж» для студентів денної форми навчання зі спеціальності 123 – «Комп'ютерна інженерія» освітнього ступеня «Магістр». Частина І

Укладачі: д. т. н., проф. М. І. Гученко,
к. т. н. П. П. Костенко,
асист. Н. Л. Сохін
Рецензент к. т. н., доц. О. Г. Славко

Кафедра комп'ютерних та інформаційних систем

Затверджено методичною радою Кременчуцького національного університету імені Михайла Остроградського
Протокол № _____ від _____
Голова методичної ради _____ проф. В. В. Костін

ЗМІСТ

ВСТУП

Методичні вказівки укладені на підставі робочої програми навчального курсу «Інформаційна стійкість комп'ютерних технологій та мереж» для студентів денної форми навчання зі спеціальності 123 – «Комп'ютерна інженерія» освітнього ступеня «Магістр».

Виконання лабораторних робіт дозволить студентам набути практичних навичок у галузі створення комп'ютерних технологій та мереж, стійких до дії негативних антропогенних та фізичних впливів, і використовувати сучасні інструментальні програмні засоби для їх проектування.

У результаті виконання лабораторних робіт студент повинен уміти:

– організовувати процес проектування та розроблення відмовостійкого програмного забезпечення;

– застосовувати методи забезпечення якості програмного забезпечення, її оцінювання та контролю;

– застосовувати методи забезпечення якості обслуговування користувачів (QoS);

– застосовувати методи керування потоками даних у комп'ютерних мережах;

– аналізувати загрози та застосовувати стандарти, що стосуються інформаційної безпеки комп'ютерних систем та мереж.

1 ПЕРЕЛІК ЛАБОРАТОРНИХ РОБІТ

Лабораторна робота № 1

Тема. Надійність апаратного забезпечення

Мета: отримання практичних навичок розрахунку надійності апаратного забезпечення комп'ютерних інформаційних систем.

Короткі теоретичні відомості

Кількісні характеристики надійності

Напрацювання – тривалість або обсяг роботи об'єкта. Для невідновлюваних і відновлюваних виробів поняття напрацювання розрізняється: у першому випадку йдеться про напрацювання до першої відмови (вона ж остання), у другому – проміжок між двома сусідніми в часі відмовами (після кожної відмови проводиться відновлення працездатності). Математичне сподівання випадкової напрацювання T:

$$M[T] = \int_0^\infty t f(t) dt = T_0 \tag{1}$$

Імовірність безвідмовної роботи – імовірність того, що в межах заданого напрацювання *t* відмова об'єкта не виникне:

$$p(t) = Bep(T \geq t) = 1 - F(t) = 1 - \int_0^t f(t) dt \tag{2}$$

Імовірність протилежної події називають *імовірністю відмови*:

$$g(t) = Bep(T \leq t) = 1 - p(t) = F(t) \tag{3}$$

Інтенсивністю відмов називають умовну щільність імовірності виникнення відмови виробу за умови, що до моменту *t* відмова не виникла:

$$\lambda(t) = \frac{f(t)}{p(t)} = -\frac{1}{p(t)} \cdot \frac{dp(t)}{dt}; \quad p(t) = \exp\left[-\int_0^t \lambda(t) dt \right]. \tag{4}$$

Функції *f (t)* і λ*(t)* вимірюють у $г^{-1}$.

Структурно-логічний аналіз автоматичних систем

Метою розрахунку надійності автоматичних систем (АС) є оптимізація конструктивних рішень і параметрів, режимів експлуатації.

Для розрахунку надійності систему розділяють на елементи. Поділ АС на елементи умовний та залежить від постановки задачі розрахунку надійності. Наприклад, під час аналізу працездатності технологічної лінії її елементами можуть вважатися окремі установки і верстати, транспортні і завантажувальні пристрої і т. д. Верстати та пристрої також можуть вважатися технічними системами і під час оцінювання їх надійності мають бути розділені на елементи – вузли, блоки. Для розрахунків параметрів надійності зручно використовувати *структурно-логічні схеми надійності* АС, які графічно відображають взаємозв'язок елементів і їх вплив на працездатність всієї системи. Структурно-логічна схема являє собою сукупність ланок, з'єднаних одна з одною послідовно або паралельно. При *послідовному* з'єднанні відмова будь-якого елемента призводить до відмови всієї системи. При *паралельному* (з точки зору надійності) з'єднанні відмова будь-якого елемента не призводить до відмови системи, доки не відмовлять всі з'єднані елементи.

Прикладом послідовного з'єднання елементів структурно-логічної схеми може бути технологічна лінія для переробки сировини в готовий продукт. Якщо ж на деяких ділянках лінії передбачена одночасна обробка на кількох однакових одиницях обладнання, то такі елементи (одиниці обладнання) можуть вважатися з'єднаними паралельно.

Розрахунки структурної надійності систем

Системи з послідовним з'єднанням елементів. Таке поєднання елементів в техніці зустрічається найбільш часто, тому його називають *основним з'єднанням*. В системі з послідовним з'єднанням для безвідмовної роботи протягом деякого напрацювання *t* необхідно і досить, щоб кожен з її ***n*** елементів працював безвідмовно протягом цього напрацювання. Вважаючи відмови елементів незалежними, імовірність одночасної безвідмовної роботи ***n*** елементів визначають за теоремою множення імовірностей: *імовірність спільної появи незалежних подій дорівнює добутку імовірностей цих подій:*

$$P(t) = p_1(t)p_2(t)...p_n(t) = \prod_{i=1}^{n} p_i(t) = \prod_{i=1}^{n}(1 - q_i(t)) \qquad (5)$$

Аргумент (*t),* що показує залежність показників надійності від часу, опускаємо для скорочення записів формул. Надійність системи з послідовним з'єднанням виявляється тим нижчою, чим більше число елементів. Окрім того, імовірність безвідмовної роботи АС з послідовним з'єднанням не може бути вище імовірності безвідмовної роботи найненадійнішого з її елементів (принцип «гірше гіршого») і з малонадійних елементів не можна створити високонадійної АС з послідовним з'єднанням.

Якщо всі елементи системи працюють в періоді нормальної експлуатації, можна записати:

$$P = \prod_{i=1}^{n} \exp(-\lambda_i t) = \exp\left[-\left(\sum_{i=1}^{n} \lambda_i\right) t\right] = \exp(-\lambda t), \qquad (6)$$

де: $\lambda = \lambda_1 + \lambda_2 + ... + \lambda_n = \sum_{i=1}^{n} \lambda_i = const$ – інтенсивність відмов системи.

Отже, інтенсивність відмов системи з послідовним з'єднанням елементів дорівнює сумі інтенсивностей відмов елементів.

Системи з паралельним з'єднанням елементів. Для відмови системи з паралельним з'єднанням елементів протягом напрацювання t необхідно і достатньо, щоб всі її елементи відмовили протягом цього напрацювання. Відмова системи полягає в спільній відмові всіх елементів, імовірність чого (у разі незалежності відмов) може бути знайдена за теоремою множення імовірностей як добуток імовірностей відмови елементів:

$$Q = q_1 q_2 ... q_n = \prod_{i=1}^{n} q_t = \prod_{i=1}^{n} (1 - p_i). \quad (7)$$

Відповідно, імовірність безвідмовної роботи

$$P = 1 - Q = 1 - \prod_{i=1}^{n} q_i = 1 - \prod_{i=1}^{n} (1 - p_i). \quad (8)$$

Імовірність відмови системи не може бути вище імовірності найнадійнішого її елемента («краще кращого») і навіть з порівняно ненадійних елементів можлива побудова цілком надійної системи.

За експоненційним розподілом напрацювання: $P = 1 - [1 - \exp(-\lambda t)]^n$, звідки середнє напрацювання системи $T_0 = \frac{1}{\lambda} \sum_{i=1}^{n} \frac{1}{i} = T_{0i} \sum_{i=1}^{n} \frac{1}{i}$,

Системи типу «m з n». Систему типу «m з n» можна розглядати як варіант системи з паралельним з'єднанням елементів, відмова якої відбудеться, якщо з n елементів, з'єднаних паралельно, працездатними виявляться менше m елементів (m < n).

Системи типу «m з n» найбільш часто зустрічаються у зв'язкових системах автоматизації технологічних ліній (при цьому елементами є сполучні

канали). Для розрахунку надійності систем типу «m з n» з порівняно невеликою кількістю елементів можна скористатися *методом прямого перебору*. Він полягає у визначенні працездатності кожного з можливих станів системи, які визначаються різними поєднаннями працездатних і непрацездатних станів елементів.

Розрахунок надійності системи «m з n» може проводитися *комбінаторним методом*, в основу якого покладено формулу біноміального розподілу. Біноміальному розподілу підпорядковується дискретна випадкова величина k – число появ деякої події в серії з n дослідів, якщо в окремому досліді імовірність появи події становить p. При цьому імовірність появи події дорівнює k і визначається:

$$P_k = C_n^k p^k (1-p)^{n-k}, \tag{12}$$

де C_n^k – біноміальний коефіцієнт, що називається «числом сполучень по «k з n» (тобто скількома різними способами можна реалізувати ситуацію «k з n»):

$$C_n^k = \frac{n!}{k!(n-k)!}. \tag{13}$$

Оскільки для відмови системи «m з n» досить, щоб кількість справних елементів було менше m, імовірність відмови може бути знайдена за теоремою додавання імовірностей для *k = 0, 1, ... (m-1)*:

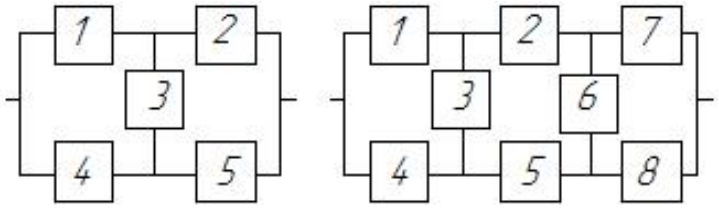
$$Q = \sum_{k=0}^{m-1} P_k = \sum_{k=0}^{m-1} C_n^k p^k (1-p)^{n-k}. \tag{14}$$

Аналогічно можна знайти імовірність безвідмовної роботи як суму для *k = m, m + 1, ..., n*:

$$P = \sum_{k=m}^{n} P_k = \sum_{k=m}^{n} C_n^k p^k (1-p)^{n-k}. \qquad (15)$$

Очевидно, що *Q + P = 1*, тому в розрахунках слід вибирати ту з формул, яка в даному конкретному випадку містить меншу кількість доданків.

Місткові схеми. Місткова структура (рис. 1 а, 1 б) не зводиться до паралельного або послідовного типу з'єднання елементів, а являє собою паралельне з'єднання послідовних ланцюжків елементів з *діагональними* елементами, включеними між вузлами різних паралельних гілок (елемент 3 на рис. 1 а, елементи 3 і 6 на рис. 1 б). Працездатність такої системи визначається не тільки кількістю елементів, що відмовили, але і їх становищем у структурній схемі. Наприклад, працездатність АС, схема якої наведена на рис. 1 а, буде втрачена, якщо одночасно відмовлять елементи 1 і 2, або 4 і 5, або 2, 3 і 4 і т. д. У той же час відмова елементів 1 і 5, або 2 і 4, або 1, 3 і 4, або 2, 3 і 5 до відмови системи не призводить.



а) б)

Рисунок 1 – Місткові системи

Для розрахунку надійності місткових систем можна скористатися *методом прямого перебору*, як це було зроблено для систем «m з n», але під час аналізу працездатності кожного стану системи необхідно враховувати не тільки число відмовили елементів, але і їх положення в схемі.

Імовірність безвідмовної роботи системи визначається як сума імовірностей всіх працездатних станів.

Для аналізу надійності АС, структурні схеми яких не зводяться до паралельного або послідовного типу, можна скористатися також *методом логічних схем із застосуванням алгебри логіки* (булевої алгебри). Застосування цього методу зводиться до складання для АС формули алгебри логіки, яка визначає умову працездатності системи. При цьому для кожного елемента і системи загалом розглядають дві протилежні події – відмова і збереження працездатності.

Для складання логічної схеми можна скористатися двома методами: мінімальних шляхів і мінімальних перетинів.

Для розрахунку верхньої межі імовірності безвідмовної роботи системи слугує *метод мінімальних перетинів*. *Мінімальним перетином* називають набір непрацездатних елементів, відмова яких призводить до відмови системи, а відновлення працездатності будь-якого з них – до відновлення працездатності системи. Мінімальних перетинів може бути кілька. Система з паралельним з'єднанням елементів має тільки один мінімальний перетин, що включає всі її елементи (відновлення будь-якого відновить працездатність системи). В системі з послідовним з'єднанням елементів число мінімальних шляхів збігається з числом елементів, і кожен перетин включає один з них.

У деяких випадках для аналізу надійності АС вдається скористатися *методом розкладання відносно особливого елемента*, що грунтується на відомій в математичній логіці теоремі про розкладанні функції логіки з будь-якого аргументу. Відповідно до неї, можна записати:

$$P = p_i P(p_i = 1) + q_i P(p_i = 0), \qquad (16)$$

де p_i та $q_i = 1-p_i$ – імовірності безвідмовної роботи і відмови i-го елемента, $P\ (p_i = 1)$ та $P\ (p_i = 0)$ – можливість працездатного стану системи за умови, що i-й елемент абсолютно надійний і що i-й елемент відмовив.

Комбіновані системи. Більшість реальних АС має складну *комбіновану структуру,* частина елементів якої утворює послідовне з'єднання, інша частина – паралельне, окремі гілки елементів або гілки структури утворюють місткову схему або типу «m з n».

Метод прямого перебору для таких систем практично не реалізується. Більш доцільно в цих випадках попередньо зробити декомпозицію системи, розбивши її на прості підсистеми – групи елементів, методика розрахунку надійності яких відома. Потім ці підсистеми в структурній схемі надійності замінюються квазіелементом з вірогідністю безвідмовної роботи, рівними обчисленими імовірностями безвідмовної роботи цих підсистем. При необхідності таку процедуру можна виконувати кілька разів.

Підвищення надійності технічних систем

Методи підвищення надійності. Розрахункові залежності для визначення основних характеристик надійності ТЗ показують, що надійність системи залежить від її структури (структурно-логічної схеми) і надійності елементів.Тому для складних систем можливі два варіанти підвищення надійності: підвищення надійності елементів і зміна структурної схеми.

Підвищення надійності елементів на перший погляд видається найбільш простим способом підвищення надійності системи. Дійсно, теоретично завжди можна вказати такі характеристики надійності елементів, щоб імовірність безвідмовної роботи системи відповідала заданим вимогам. Однак практична реалізація такої високої надійності елементів може перестати працювати. Розгляд методів забезпечення надійності елементів АС є предметом спеціальних технологічних і фізико-хімічних дисциплін і виходить за межі теорії надійності. Однак, у будь-якому разі, високонадійні елементи зазвичай мають великі габарити, масу та вартість. Виняток становить використання більш досконалої елементної бази, яка реалізується на принципово нових фізичних і технологічних принципах.

Зміна структури системи для підвищення надійності передбачає два аспекти. З одного боку, це означає перебудову конструктивної або

функціональної схеми АС (структури зв’язків між складовими елементами), зміну принципів функціонування окремих частин системи (наприклад, перехід від аналогової обробки сигналів до цифрової). Такі перетворення трапляються рідко, так що цей прийом, загалом, не вирішує проблеми надійності.

З іншого боку, зміна структури розуміється як введення до АС додаткових, надлишкових елементів, що включаються в роботу у разі відмови основних. Застосування додаткових коштів і можливостей з метою збереження працездатного стану об’єкта у разі відмови одного або декількох його елементів називають резервуванням.

Принцип резервування подібний до розглянутого раніше паралельного з’єднання елементів і з’єднання типу «n з m», де завдяки надмірності можливе забезпечення більш високої надійності системи, ніж її елементів.

Виділяють кілька видів резервування: тимчасове, інформаційне, функціональне та ін. Щодо аналізу структурної надійності інтерес викликає структурне резервування – введення до структури об’єкта додаткових елементів, що виконують функції основних елементів в разі їх відмови.

Класифікація різних способів структурного резервування здійснюється за такими ознаками:

1) за схемою включення резерву:

– загальне резервування (резервується об’єкт загалом);

– роздільне резервування (резервуються окремі елементи або їх групи);

– змішане резервування (різні види резервування поєднуються в одному об’єкті);

2) за способом включення резерву:

– постійне резервування, без перебудови структури об’єкта у разі виникнення відмови його елемента;

– динамічне резервування (у разі відмови елемента відбувається перебудова структури схеми). У свою чергу підрозділяється на:

а) резервування заміщенням (функції основного елемента передаються резервному тільки після відмови основного);

б) ковзне резервування (кілька основних елементів резервується одним або декількома резервними, кожен з яких може замінити будь-який основний (тобто групи основних і резервних елементів ідентичні);

3) за станом резерву:

– навантажене резервування (резервні елементи (або один з них) знаходяться в режимі основного елемента);

– полегшене резервування (резервні елементи (чи хоча б один з них) знаходяться в менш навантаженому режимі порівняно з основними);

– ненавантажене резервування (резервні елементи до початку виконання ними функцій знаходяться в ненавантаженому режимі).

Основною характеристикою структурного резервування є кратність резервування – відношення числа резервних елементів до числа зарезервованих ними основних елементів, виражене нескорочуваним дробом (типу 2:3; 4:2 і т. д.). Резервування одного основного елементу одним резервним (тобто з кратністю 1:1) називають дублюванням.

Кількісне підвищення надійності системи в результаті резервування або застосування високонадійних елементів можна оцінити за коефіцієнтом виграшу надійності, що визначається як відношення показника надійності до і після перетворення системи.

Розглянемо приклад розрахунку надійності АС. Структурна схема надійності наведена на рис. 2. Значення інтенсивності відмов елементів подані в 10^{-6} 1/г.

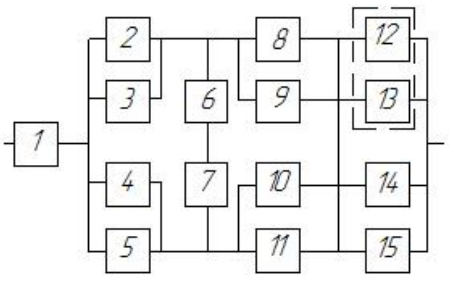


Рисунок 2 – Початкова структурна схема надійності

$\lambda_1 = 0{,}001;$

$\lambda_2 = \lambda_3 = \lambda_4 = \lambda_5 = 0{,}1;$

$\lambda_6 = \lambda_7 = 0{,}01;$

$\lambda_8 = \lambda_9 = \lambda_{10} = \lambda_{11} = 0{,}2;$

$\lambda_{12} = \lambda_{13} = \lambda_{14} = \lambda_{15} = 0{,}5;$

$\gamma = 50\ \%.$

1. У вихідній схемі елементи 2 і 3 утворюють паралельне з'єднання. Замінюємо їх квазіелементом A:

$$p_A = 1 - q_2 q_3 = 1 - q_2^2 = 1 - (1 - p_2)^2.$$

2. Елементи 4 і 5 також утворюють паралельне з'єднання, замінивши яке елементом B і з огляду на те, що $p_4 = p_5 = p_6$, отримаємо:

$$p_B = 1 - q_4 q_5 = 1 - q_2^2 = p_A.$$

3. Елементи 6 і 7 у вихідній схемі з'єднані послідовно. Замінюємо їх елементом C, для якого за $p_6 = p_7$:

$$p_C = p_6 p_7 = p_6^2.$$

4. Елементи 8 і 9 утворюють паралельне з'єднання. Замінюємо їх елементом D, для якого за $p_8 = p_9$, отримаємо:

$$p_D = 1 - q_8 q_9 = 1 - q_8^2 = 1 - (1 - p_8)^2.$$

5. Елементи 10 і 11 з паралельним з'єднанням замінюємо елементом E, до того ж оскільки $p_{10} = p_{11} = p_8$, то:

$$p_E = 1 - q_{10} q_{11} = 1 - q_{10}^2 = 1 - (1 - p_{10})^2 = p_D.$$

6. Елементи 12, 13, 14 і 15 утворюють з'єднання «2 з 4», яке замінюємо елементом F. Оскільки $p_{12} = p_{13} = p_{14} = p_{15}$, то для визначення імовірності безвідмовної роботи елемента F можна скористатися комбінаторним методом:

$$P_E = \sum_{k=2}^{4} p_k = \sum_{k=2}^{4} C_4^k p_{12}^k (1 - p_{12})^{4-k} = \frac{4!}{2!2!} p_{12}^2 (1 - p_{12})^2 + \frac{4!}{3!1!} p_{12}^3 (1 - p_{12}) +$$
$$+ \frac{4!}{4!0!} p_{12}^4 = 6 p_{12}^2 (1 - p_{12})^2 + 4 p_{12}^3 (1 - p_{12}) + p^4 = 6 p_{12}^2 - 8 p_{12}^3 + 3 p_{12}^4.$$

7. Змінена схема зображена на рис. 3.

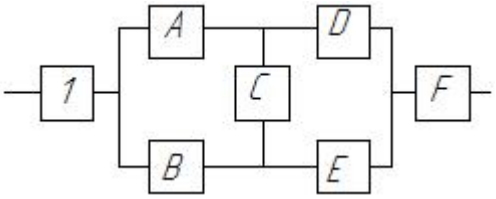


Рисунок 3 – Змінена схема

8. Елементи A, B, C, D і E утворюють місткову систему (рис. 3), яку можна замінити квазіелементом G. Для розрахунку імовірності безвідмовної роботи скористаємося методом розкладання щодо особливого елемента, в якості якого виберемо елемент C. Тоді:

$$p_G = p_C p_G(p_C = 1) + q_C p_C(p_C = 0),$$

де $p_G\ (p_C = 1)$ – імовірність безвідмовної роботи місткової схеми з абсолютно надійним елементом C (рис. 4 а), $p_G\ (p_C = 0)$ – імовірність безвідмовної роботи місткової схеми при відмові елемента C (рис.4 б).

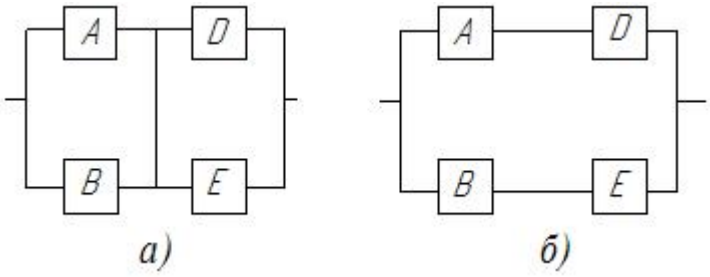


Рисунок 4 – Перетворення місткової схеми з абсолютно надійним (а) і таким, що відмовив (б) елементом C

З огляду на те, що $p_B = p_A$, отримаємо:

$$p_G = p_C\left[1-(1-p_A)(1-p_B)\right]\cdot\left[1-(1-p_D)(1-p_E)\right]+$$
$$+(1+p_C)\left[1-(1-p_Ap_B)(1-p_Dp_E)\right]=$$
$$=p_C\left[1-(1-p_A)^2\right]\cdot\left[1-(1-p_D)^2\right]+(1-p_C)\left[1-(1-p_A^2)(1-p_D^2)\right]=$$
$$=p_C(2p_A-p_A^2)(2p_D-p_D^2)+(1-p_C)(p_A^2+p_D^2-p_A^2p_D^2)=$$
$$=p_Ap_Cp_D(2-p_A)(2-p_D)+(1-p_C)(p_A^2+p_D^2-p_A^2p_D^2).$$

9. Після перетворень схема зображена на рис. 5.

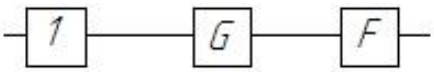


Рисунок 5 – Змінена схема

10. У зміненій схемі (рис. 5) елементи 1, G та F утворюють послідовне з'єднання. Тоді імовірність безвідмовної роботи всієї системи $P = p_1 p_G p_F$.

11. Оскільки за умовою всі елементи системи працюють в періоді нормальної експлуатації, то імовірність безвідмовної роботи елементів 1–15 описується експоненціальним законом $P_i = \exp^{(-\lambda_i t)}$.

12. Результати розрахунків імовірностей безвідмовної роботи елементів 1–15 вихідної схеми для напрацювання до 3×10^6 годин представлені в табл. 1.

13. Результати розрахунків імовірностей безвідмовної роботи квазіелементів A, B, C, D, E, F і G також подані в табл. 1.

Таблиця 1 – Розрахунок імовірності безвідмовної роботи системи

Елемент	**λ_i, x10^{-6} 1/г$^{-1}$**	**Напрацювання t, x 10 ^{6} г**							
		0,5	**1,0**	**1,5**	**2,0**	**2,5**	**3,0**	**1,9**	**2,85**
1	0,001	0,9995	0,9990	0,9985	0,9980	0,9975	0,9970	0,9981	0,9972
2-5	0,1	0,9512	0,9048	0,8607	0,8187	0,7788	0,7408	0,8270	0,7520
6,7	0,01	0,9950	0,9900	0,9851	0,9802	0,9753	0,9704	0,9812	0,9719
8-11	0,2	0,9048	0,8187	0,7408	0,6703	0,6065	0,5488	0,6839	0,5655
12-15	0,5	0,7788	0,6065	0,4724	0,3679	0,2865	0,2231	0,3867	0,2405
A, B	-	0,9976	0,9909	0,9806	0,9671	0,9511	0,9328	0,9701	0,9385
C	-	0,9900	0,9801	0,9704	0,9608	0,9512	0,9417	0,9628	0,9446
D, E	-	0,9909	0,9671	0,9328	0,8913	0,8452	0,7964	0,9001	0,8112
F	-	0,9639	0,8282	0,6450	0,4687	0,3245	0,2172	0,5017	0,2458
G	-	0,9924	0,9888	0,9863	0,9820	0,9732	0,9583	0,9832	0,9594
P	-	0,9561	0,8181	0,6352	0,4593	0,3150	0,2075	0,4923	0,2352
12`-15`	0,322	0,8513	0,7143	0,6169	0,5252	0,4471	0,3806	0,5424	0,3994
F`	-	0,9883	0,9270	0,8397	0,7243	0,6043	0,4910	0,7483	0,5238
P`	-	0,9803	0,9157	0,8270	0,7098	0,5866	0,4691	0,7343	0,5011
16-18	0,5	0,7788	0,6065	0,4724	0,3679	0,2865	0,2231	0,3867	0,2405

Продовження таблиці 1

Еле-мент	λ_i, $x10^{-6}$ $1/г^{-1}$	Напрацювання t, х 10^6 г							
		0,5	1,0	1,5	2,0	2,5	3,0	1,9	2,85
F``	-	0,9993	0,9828	0,9173	0,7954	0,6413	0,4858	0,8233	0,5311
P``	-	0,9912	0,9708	0,9034	0,7795	0,6226	0,4641	0,8079	0,5081

Порядок виконання роботи

1. Отримати у викладача варіант завдання структурної схеми АС.

2. Провести структурні перетворення вихідної схеми.

3. Заповнити таблицю розрахунку імовірності безвідмовної роботи системи.

4. Побудувати графік зміни імовірності безвідмовної роботи системи від часу напрацювання.

Зміст звіту

1. Назва та мета роботи.

2. Опис виконання роботи.

3. Письмові відповіді на контрольні запитання.

Контрольні питання

1. Надайте визначення поняття «надійність». Необхідність забезпечення надійності компонентів інформаційних систем.

2. Основні показники надійності.

3. Надайте визначення поняття «відмова». Види і причини відмов і пошкоджень обладнання.

4. Основні показники теорії імовірностей і математичної статистики, які використовують у теорії надійності.

5. Надійність обладнання в період нормальної експлуатації. Який закон використовують під час розрахунку надійності? Його переваги.

6. Основні показники, що характеризують надійність відновлюваних виробів.

7. Як визначають імовірність безвідмовної роботи системи?

8. Основні положення методу розрахунку надійності послідовної системи.

9. Основні способи підвищення надійності пристроїв.

Література: [1–4, 13].

Лабораторна робота № 2

Тема. Проектування відмовостійкого програмного забезпечення

Мета: отримання практичних навичок розробки програмного забезпечення, стійкого до відмов.

Короткі теоретичні відомості

З чотирьох основних груп методів забезпечення надійності програмного забезпечення найрезультативнішими є методи попередження помилок. У більшості випадків, однак, під час розробки програмного забезпечення ніяк не можна припускати, що готова програма не міститиме помилок. У цьому і полягає вихідна передумова методів виявлення помилок, виправлення помилок і забезпечення стійкості до помилок: готова програма (система) міститиме помилки і тому має бути спроектована так, щоб її поведінка було передбачуваною і у разі помилки. Це стосується як помилок у програмному забезпеченні, так і помилок користувача або збоїв апаратури.

Пасивне виявлення помилок

Якщо припускати, що у програмному забезпеченні будуть помилки, то, очевидно, в першу чергу слід вжити заходів для їх виявлення. Крім того, якщо необхідно вживати додаткових заходів (наприклад, виправляти помилки або їх наслідки), то все одно спочатку потрібно вміти виявляти помилки.

Заходи з виявлення помилок можна поділити на дві підгрупи: пасивні спроби виявити симптоми помилки під час «звичайної» роботи програмного забезпечення та активні спроби програмної системи періодично обстежувати свій стан в пошуках ознак помилок.

Заходи з виявлення помилок можуть бути прийняті на декількох структурних рівнях програмної системи. Розглянемо рівень підсистем, або компонент, та заходи з виявлення симптомів помилок, що виникають під час переходу від однієї компоненти до іншої, а також всередині компоненти. Все це, звичайно, застосовують також до окремих модулів всередині компоненти.

Розробляючи ці заходи, слід враховувати такі положення:

1) взаємна недовіра. Кожна з компонент має припускати, що всі інші містять помилки. Коли вона отримує дані від іншої компоненти або з джерела поза системою, вона повинна припускати, що дані можуть бути неправильними, і намагатися знайти в них помилки;

2) негайне виявлення. Помилки необхідно виявити якомога раніше. Це не тільки обмежує збиток, але і значно спрощує завдання налагодження;

3) надмірність. Всі засоби виявлення помилок грунтуються на деякій формі надмірності (явній або неявній).

Конкретні заходи з виявлення залежать від специфіки прикладної області. Ось деякі з них:

1) перевіряти атрибути будь-якого елемента вхідних даних. Якщо вхідні дані мають бути числовими або літерними, слід перевірити це. Якщо число на вході має бути позитивним, слід перевірити його значення. Якщо відомо, якою має бути довжина вхідних даних, слід перевірити її;

2) застосовувати «теги» в таблицях, записах і керуючих блоках і перевіряти їх з допомогою допустимості вхідних даних. Тег – це поле запису, що явно вказує на його призначення;

3) перевіряти, чи знаходиться вхідне значення у встановлених межах. Наприклад, якщо вхідний елемент – адреса в основній пам’яті, слід перевірити його допустимість. Завжди перевіряти поле адреси або покажчика на нуль і вважати, що воно неправильне, якщо дорівнює нулю. Якщо вхідні дані – таблиця імовірностей, слід перевірити, чи знаходяться всі значення між нулем і одиницею;

4) перевіряти допустимість всіх варіантів значень. Якщо вхідне поле –

код, що позначає один з десяти районів, ніколи не слід припускати, що якщо це не код жодного з районів 1, 2, ..., 9, то це обов'язково код району 10;

5) якщо у вхідних даних є будь-яка явна надмірність, слід скористатися нею для перевірки даних;

6) там, де у вхідних даних немає явної надмірності, необхідно ввести її. Якщо система використовує вкрай важливу таблицю, варто включити до неї контрольну суму. Щоразу, коли таблиця оновлюється, слід підсумувати (за деяким модулем) її поля і результат помістити в спеціальне поле контрольної суми. Підсистема, що використовує таблицю, зможе тепер перевірити, чи не була таблиця випадково зіпсована, – для цього потрібно виконати контрольне підсумовування;

7) порівнювати, чи узгоджуються вхідні дані з якими-небудь внутрішніми даними. Якщо на вході операційної системи виникає вимога звільнити деякий блок пам'яті, вона повинна переконатися, що цей блок в даний момент дійсно зайнятий.

Коли розробляються заходи з виявлення помилок, важливо вибрати узгоджену стратегію для всієї системи (тобто застосувати ідею концептуальної цілісності до виявлення помилок). Дії після виявлення помилки в програмному забезпеченні (наприклад, повернення коду помилки) повинні бути одноманітними для всіх компонент системи.

Активне виявлення помилок

Не всі помилки можна виявити пасивними методами, оскільки ці методи виявляють помилку лише тоді, коли її симптоми піддаються відповідній перевірці. Можна робити і додаткові перевірки, якщо спроектувати спеціальні програмні засоби для активного пошуку ознак помилок в системі. Такі засоби називаються засобами активного виявлення помилок.

Активні засоби виявлення помилок зазвичай об'єднуються в діагностичний монітор: паралельний процес, який періодично аналізує стан системи для виявлення помилки.

Діагностичний монітор можна реалізувати як періодично виконувану

задачу (наприклад, вона планується на кожну годину) або як задачу з низьким пріоритетом, яка планується для виконання в той час, коли система переходить в стан очікування. Виконувані монітором конкретні перевірки залежать від специфіки системи.

Монітор може обстежити основну пам'ять, щоб виявити блоки пам'яті, що не виділені жодній з виконуваних задач та не включені до системного списку вільної пам'яті. Він може перевіряти також незвичайні ситуації: наприклад, процес не планувався для виконання протягом деякого інтервалу часу.

Монітор може здійснювати пошук усередині системи повідомлень або операцій введення-виведення, які незвично довгий час залишаються незавершеними, ділянок пам'яті на диску, що не позначені як виділені і не включені до списку вільної пам'яті, а також різних відхилень у файлах даних.

Виправлення помилок і стійкість до помилок

Маючи засоби виявлення помилок у програмному забезпеченні, наступним кроком є створення засобів, націлених на виправлення виявлених помилок. Термін «виправлення помилок» у застосуванні до програмного забезпечення означає ліквідацію наслідків, викликаних помилкою, а не виправлення самої помилки.

Ізоляція помилок

У великій обчислювальній системі ізоляція програм є ключовим чинником, який гарантує, що відмови в програмі одного користувача не призведуть до відмов у програмах інших користувачів або до повного виведення системи з ладу. Основні правила ізоляції помилок:

1) прикладна програма не повинна мати можливості безпосередньо посилатися на іншу прикладну програму або дані в іншій програмі і змінювати їх;

2) прикладна програма не повинна мати можливості безпосередньо посилатися на програми або дані операційної системи і змінювати їх. Зв'язок між двома програмами (або програмою і операційною системою) може бути

дозволено тільки за умови використання чітко визначених сполучень і тільки у випадку, коли обидві програми надають згоду на цей зв’язок;

3) прикладні програми та їх дані повинні бути захищені від операційної системи до такої міри, щоб помилки в операційній системі не могли призвести до випадкової зміни прикладних програм або їх даних;

4) операційна система має захищати всі прикладні програми і дані від випадкової їх зміни операторами системи або обслуговуючим персоналом;

5) прикладні програми не повинні мати можливості ні зупинити систему, ні змусити її змінити іншу прикладну програму або її дані;

6) коли прикладна програма звертається до операційної системи, повинна перевірятися допустимість всіх параметрів. Окрім того, прикладна програма не повинна мати можливості змінити ці параметри між моментами перевірки і реального їх використання операційною системою;

7) ніякі системні дані, безпосередньо доступні прикладним програмам, не повинні впливати на функціонування операційної системи;

8) прикладні програми не повинні мати можливості в обхід операційної системи прямо використовувати керовані нею апаратні ресурси. Прикладні програми не повинні прямо викликати компоненти операційної системи, призначені для використання тільки її підсистемами;

9) компоненти операційної системи повинні бути ізольовані один від одного так, щоб помилка в одній з них не призвела до зміни інших компонент або їх даних;

10) якщо операційна система виявляє помилку в собі самій, вона повинна спробувати обмежити вплив цієї помилки однією прикладною програмою і в крайньому випадку припинити виконання тільки цієї програми;

11) операційна система повинна давати прикладним програмам можливість на вимогу виправляти виявлені в них помилки, а не беззастережно припиняти їх виконання.

Порядок виконання роботи

1. Вибрати один з варіантів програмних модулів:

– розв’язання квадратного рівняння;

– площа трикутника;

– інтерполяційний поліном лагранжа;

– сортування списку;

– інтегрування функції;

– редактор рядка;

– сума ряду;

– скільки разів в матриці зустрічається задане число;

– середнє арифметичне всіх парних елементів масиву, що стоять на непарних місцях;

– сума елементів частин масиву;

– куби чисел від a до b;

– додавання правильного закінчення (слова) до числа;

– обчислення факторіала числа;

– приклад форматованого виведення нематеріальних типів;

– приклад найпростішого введення і виведення даних;

– приклад форматованого виведення дійсних чисел;

– визначити кількість простих чисел в масиві;

– перевірка кратності числа;

– приклад використання запису з варіантами;

– фільтрація записів за значенням поля;

– сума та добуток цифр числа;

– квадрати натуральних чисел;

– видалення однакових символів у рядку;

– частота народження символу в рядку;

– переворот рядка;

– заміна підрядка в рядку;

– вставка підрядка в рядку;

– видалення підрядка в рядку.

2. Розробити програму з графічним інтерфейсом, використовуючи три підгрупи способів підвищення надійності ПЗ: динамічна надмірність, методи відступу і методи ізоляції помилок.

Зміст звіту

1. Назва та мета роботи.

2. Код програми, що містить засоби виявлення, попередження та усунення помилок.

Контрольні питання

Задаються індивідуально щодо тексту програми.

Література: [5–6].

Лабораторна робота № 3

Тема. Тестування програмного забезпечення

Мета: отримання практичних навичок тестування програмного забезпечення методами, які використовуються для виявлення помилок і дефектів.

Короткі теоретичні відомості

Тестування – невід’ємна складова процесу програмної інженерії, один з методів подальшого поліпшення якості розробленого програмного забезпечення системи за допомогою виявлення дефектів, що залишилися, не виявлених раніше іншими видами перевірок.

Процес тестування складається з декількох етапів:

- тестування компонентів;
- тестування модулів;
- тестування підсистем;
- тестування системи;
- приймальні випробування.

Тестування методом «чорного ящика»

Функціональне тестування, або тестування методом «чорного ящика»

базується на тому, що всі тести грунтуються на специфікації системи або її компонентів. Система вважається «чорним ящиком», поведінку якого можна визначити тільки за допомогою вивчення її вхідних і відповідних вихідних даних. Інша назва цього методу – функціональне тестування – пов'язана з тим, що випробувач перевіряє не реалізацію ПЗ, а тільки його виконувані функції.

Області еквівалентності

Вхідні дані програм часто можна поділити на декілька класів. Вхідні дані, що належать одному класу, мають загальні властивості, наприклад, це додатні числа, від'ємні числа, рядки без пробілів тощо. Зазвичай для всіх даних з якого-небудь класу поведінка програми однакова (еквівалентна). Через це такі класи даних іноді називають областями еквівалентності. Один з систематичних методів виявлення дефектів полягає у визначенні всіх областей еквівалентності, що обробляються програмою. Контрольні тести розробляються так, щоб вхідні і вихідні дані знаходились в межах цих областей.

Структурне тестування

Метод структурного тестування припускає створення тестів на основі структури системи і її реалізації. Такий підхід іноді називають тестуванням методом «білого ящика», «скляного ящика» або «прозорого ящика», щоб відрізняти його від тестування методом чорного ящика.

Зазвичай структурне тестування застосовують до відносно невеликих програмних елементів, наприклад, до підпрограм або методів, що асоціюються з об'єктами. За такого підходу випробувач аналізує програмний код і для отримання тестових даних використовує знання про структуру компонента. Наприклад, з аналізу коду можна визначити, скільки контрольних тестів потрібно виконати для того, щоб під час тестування всі оператори виконалися принаймні один раз.

Тестування гілок

Це метод структурного тестування, за якого перевіряються всі незалежно виконувані гілки компонента або програми. Якщо виконуються всі незалежні гілки, то і всі оператори повинні виконуватися принаймні один раз. Окрім того,

всі умовні оператори тестуються як з дійсними, так і з помилковими значеннями умов. У об’єктно-орієнтованих системах тестування гілок використовують для тестування методів, що асоціюються з об’єктами.

Після того, як протестовані всі окремі програмні компоненти, виконується збірка системи, внаслідок чого створюється часткова або повна система. Процес інтеграції системи включає збірку і тестування отриманої системи, в ході якого виявляються проблеми, що виникають під час взаємодії компонентів. Тести, перевіряючи збірку системи, повинні розроблятися на основі системної специфікації, причому тестування збірки слід починати відразу після створення працездатних версій компонентів системи.

Порядок виконання роботи

1. Розробити систему тестів для тестування створеної в лабораторній роботі № 2 програми методами «чорного» та «білого ящиків».
2. Протестувати програму та занотувати результати.

Зміст звіту

1. Назва та мета роботи.
2. Опис виконання роботи.
3. Письмові відповіді на контрольні запитання.

Контрольні питання

1. Тестування методом покриття операторів.
2. Тестування методом покриття умов.
3. Тестування методом покриття рішень.
4. Тестування методом покриття рішень/умов.
5. Тестування методом комбінаторного покриття умов.
6. Тестування методом аналізу граничних значень.
7. Тестування методом еквівалентного роздроблення.
8. Тестування методом функціональних діаграм.
9. Тестування методом припущення про помилку.
10. Метод висхідного тестування.

Література: [7–9, 13].

Лабораторна робота № 4

Тема. Керування потоками даних у комп'ютерних мережах

Мета: отримання практичних навичок захисту локальних комп'ютерних мереж за допомогою моніторингу.

Короткі теоретичні відомості

Утиліта TCPView

TCPView – це утиліта, призначена для ОС Windows, яка виводить на екран списки кінцевих вузлів всіх установлених у системі з'єднань за протоколами *TCP* і *UDP* з докладними даними, зокрема із вказівкою локальних і віддалених адрес і стану TCP-з*'єднань*, повідомляє ім'я процесу, якому належить кінцевий вузол. Під час запуску утиліта *TCPView* перераховує всі з'єднання *TCP* і *UDP*, конвертує всі IP-адреси в доменні назви.

Утиліта *TCPView* є доповненням утиліти *Netstat*, що поставляється разом з ОС Windows. Вона надає розширений набір відомостей у більш зручній формі. У комплект завантаження утиліти *TCPView* входить програма *Tcpvcon* з тими ж функціональними можливостями, але призначена для роботи в режимі командного рядка.

Сканер безпеки Xspider

XSpider – це засіб мережевого аудита, призначений для пошуку уразливостей на серверах і робочих станціях. *XSpider* дозволяє виявляти уразливості на комп'ютерах, що працюють під керуванням різних операційних систем: AIX, Solaris, Unix-Системи, Windows і інші. Програма працює під керуванням MS Windows 95/98/Millenium/NT/2000/XP/.NET.

XSpider може в автоматичному режимі перевіряти комп'ютери й сервіси в мережі на виявлення уразливості. База уразливостей постійно поповнюється фахівцями Positive Technologies. *XSpider* може виконувати перевірки за розкладом. *XSpider* виводить у звіт про результати перевірки не тільки інформацію про знайдену уразливість, але й посилання, які описують виявлену *XSpider* уразливість і дають рекомендації з її усунення. В *XSpider* версії 7, використовуються евристичні алгоритми, які не просто перебирають

уразливості з бази, але й виконують додатковий аналіз під час роботи з урахуванням особливостей поточної ситуації. Завдяки цьому, *Xspider 7* може іноді виявити специфічну уразливість, інформація про яку ще не була опублікована.

Утиліта *Network Monitor*

Network Monitor поставляється у двох версіях: спрощеній, яка включена в ОС *Windows,* і повній, що ввійшла до складу окремого продукту *Systems Management Server*. Спрощена версія може контролювати тільки локальний трафік, тобто тільки ті кадри, які приймаються мережевим адаптером робочої станції, що відслідковується.

Повна версія програми фіксує повністю весь трафік у мережі, під час підключення до іншого сервера або робочої станції, на яких установлений *Network Monitor,* і дозволяє контролювати вилучену систему.

Утиліта *Network Monitor* має фільтри, що дозволяють виділити спеціальну інформацію при великих мережевих потоках; фільтри захвата, що вибирають із всієї захопленої інформації ту, яка необхідна; і тригери, що дозволяють системі виконувати певні дії з даними, що утримуються в пакетах.

Після запуску програми вибирається мережа, трафік якої буде контролюватися. Мережа вибирається за допомогою меню *Capture* і пункту *Networks*. У вікні, що з’явиться, відображаються всі мережеві інтерфейси, які є в даному комп’ютері, мережеві адаптери, COM-порти служби RAS, якщо вона встановлена на комп’ютері.

Утиліта *Iris The Network Traffic Analyzer,* окрім стандартних функцій збору, фільтрації й пошуку пакетів, побудови звітів, має можливість реконструювання даних. *Iris The Network Traffic Analyzer* допомагає відтворити сеанси роботи користувачів з різними ресурсами.

Технологія реконструювання даних, реалізована в модулі дешифрування (*decode module*), перетворює зібрані двійкові мережеві пакети на вихідний вид, розширюючи можливості наявних засобів моніторингу й аудиту.

Аналізатор пакетів дозволяє зафіксувати різні деталі атаки, такі як дата й

час, IP-адреси й DNS-імена комп'ютерів зловмисника, а також використані порти.

Аналізатор пакетів може відтворити точну, аж до натискання клавіш і рухів миші, картину вторгнення, що необхідна для усунення наслідків атаки й посилення заходів безпеки. Аналізатор пакетів дозволить підсилити захист корпоративної мережі.

Порядок виконання роботи

1. Відкрийте утиліту *TCPView*. Під час запуску ви побачите список всіх активних кінцевих вузлів з'єднань за протоколами *TCP* і *UDP*. Ознайомтеся з основними пунктами меню. Змініть період відновлення інформації, за допомогою пункту «*Період відновлення» (Refresh Rate)* у меню «*Параметри» (Options)*. Якщо в період між відновленнями стан кінцевих вузлів мережі змінився, вони виділяються жовтим кольором, якщо вузол не знайдено, то червоним кольором, нові вузли мережі відображаються зеленим кольором.

2. Занотуйте результати сканування відкритих з'єднань.

3. Закрийте встановлені підключення за протоколами *TCP/IP* зі станом «*Установлене» (Established)* за допомогою пункту «*Закрити підключення» (Close Connections)* у меню «*Файл» (File)*. Закрийте утиліту *TCPView*.

4. Відкрийте утиліту *XSpider*. Вивчіть основні пункти меню, скориставшись документацією з меню «*Довідка*».

5. Сформуйте завдання для сканування. Для цього створіть нове робоче вікно *XSpider* і додайте список робочих станцій. Для цього виберіть команду меню «*Виправлення/Додати*». У вікні, що з'явилося, *задайте IP-адресу*, імена або діапазон комп'ютерів згідно із завданням. Обмежте одночасне сканування тільки для двох робочих станцій.

6. Створіть свій профіль сканування. Для цього виберіть пункти «*Профіль/Створення нового профілю*». Визначте набір налаштувань для сканування відповідно до завдання. Для цього в діалозі, що з'явився, у дереві налаштувань виберіть пункт *«Сканер уразливостей/Визначення уразливостей*». Збережіть створені завдання.

7. Запустіть сканування. Для цього виберіть відповідну команду з меню «*Сканування*».

8. Перескануйте окремі сервіси. Це може знадобитися в деяких особливих ситуаціях, наприклад, для підтвердження наявності DoS-уразливості. Іноді, за поганої якості зв'язку, перевіряється можливе помилкове визначення DoS-уразливості, коли зв'язок з вузлом перервався випадково, а *XSpider* зробив висновок, що пройшла DoS-атака. У цьому разі можна провести повторне сканування відповідного сервісу й за повторного виявлення уразливості більш упевнено зробити висновок про її наявність.

9. Заплануйте запуск вашого завдання за допомогою планувальника. Для цього викличте з меню «*Розклад/Створити*» майстра створення розкладу. Виберіть завдання для автоматизації зі списку збережених завдань у папці *Tasks* або натисніть «*Огляд*», якщо потрібне завдання перебуває не в папці *Tasks*. Виберіть інтервал періодичності перевірок та інтервал, протягом якого створюваний розклад буде активним. Задайте автоматичну генерацію й параметри звіту після кожного автоматичного виконання завдання.

10. За результатами кожного сканування згенеруйте звіт про поточне сканування. Для цього виберіть команду з меню «*Сервіс/Створити звіт*» у вікні «*Майстер створення звіту*». Виберіть формат звіту *RTF*, потім виберіть варіант звіту.

11. Проаналізуйте результати сканування вашого завдання. Занесіть до звіту результат сканування.

12. Занесіть до звіту порівняльну характеристику отриманих вами результатів за допомогою утиліт *TCPView* і *XSpider*.

13. Запустіть утиліту *Network Monitor*. Ознайомтеся з основними механізмами моніторингу мережевого трафіка, а також з налаштуваннями, наявними в пунктах меню *Вид (View)*, *Фрейми* (*Frames*), *Перехоплення* (*Capture*), *Фільтр* (*Filter*), *Властивості* (*Tools*).

14. Налаштуйте й запустіть перехоплення трафіка. Для цього скористайтеся вікном редактора фільтра запису (*Capture Filter*) або фільтра

перегляду (*Display Filter*), що відкривається з вікном *Capture*. Через деякий час зупиніть перехоплення заданого трафіка. Для цього у меню *Capture* виберіть меню зупинки перехоплення. Занесіть до протоколу правила фільтра, які ви будете використовувати в п. 15–38.

15. Настройте перехоплення трафіка між двома сусідніми робочими станціями.

16. Налаштуйте перехоплення трафіка між локальною робочою станцією й сусідньою.

17. Налаштуйте перехоплення за протоколом *ARP*.

18. Налаштуйте запис кадрів, що містять певний тип даних.

19. Налаштуйте перехоплення всього мережевого трафіка.

20. Налаштуйте перехоплення тільки зовнішнього трафіка.

21. Відфільтруйте кадри для відображення широкомовних пакетів у мережі.

22. Відфільтруйте кадри для відображення адреси відправника й приймача кадру для канального й мережевого рівнів.

23. Відфільтруйте кадри таким чином, щоб відобразити кадри, для відправлення яких використовувався протокол FTP.

24. Відфільтруйте кадри таким чином, щоб відобразити кадри, передані за протоколом TCP.

25. Відфільтруйте кадри таким чином, щоб відобразити кадри, передані за протоколом IP.

26. Відфільтруйте кадри таким чином, щоб відобразити весь мережевий трафік, крім широкомовних повідомлень.

27. Налаштуйте перехоплення трафіка між двома сусідніми робочими станціями.

28. Налаштуйте перехоплення трафіка між локальною робочою станцією й сусідньою.

29. Налаштуйте перехоплення за протоколом *ARP*.

30. Налаштуйте запис кадрів, що містять певний тип даних.

31. Налаштуйте перехоплення всього мережевого трафіка.

32. Налаштуйте перехоплення тільки зовнішнього трафіка.

33. Відфільтруйте кадри для відображення широкомовних пакетів у мережі.

34. Відфільтруйте кадри для відображення адреси відправника й приймача кадру для канального й мережевого рівнів.

35. Відфільтруйте кадри таким чином, щоб відобразити кадри, для відправлення яких використовувався протокол FTP.

36. Відфільтруйте кадри таким чином, щоб відобразити кадри, передані за протоколом TCP.

37. Відфільтруйте кадри таким чином, щоб відобразити кадри, передані за протоколом IP.

38. Відфільтруйте кадри таким чином, щоб відобразити весь мережевий трафік, окрім широкомовних повідомлень.

Зміст звіту

1. Назва та мета роботи.

2. Опис виконання роботи.

3. Письмові відповіді на контрольні запитання.

Контрольні питання

1. Поясніть призначення утиліт TCPView і XSpider. У чому полягає принципова відмінність цих утиліт?

2. Які фактори можуть спричинити перевантаження каналу утиліти XSpider? Як можна боротися з даним перевантаженням?

3. Які типи уразливостей можна виявити за допомогою утиліти XSpider?

4. За портами яких мережевих служб ведеться спостереження за допомогою сканера безпеки XSpider?

5. Поясніть призначення сервісу «Перевизначення уразливості».

6. Де зберігається, за замовчуванням, файл портів утиліти XSpider і як можна налаштувати сканування тільки за певними портами?

7. Поясніть поняття евристичних методів пошуку уразливостей. У чому

перевага цього методу?

8. Як можна автоматизувати роботу утиліти XSpider?

9. На якому рівні семирівневої моделі OSI проводиться збір даних в утиліті Network Monitor?

10. Яку інформацію про передані й отримані мережею дані бачить адміністратор за допомогою утиліти Network Monitor?

11. Для чого слугують і як налаштовуються фільтр запису та фільтр відображення утиліти Network Monitor?

12. Як можна виявити утиліту Network Monitor і яка інформація видається у разі її виявлення?

13. Поясніть, як можна одержати інформацію про зловмисника за допомогою утиліти Network Monitor?

Література: [10–13].

2 КРИТЕРІЇ ОЦІНЮВАННЯ ЗНАНЬ СТУДЕНТІВ

У 10-му семестрі студенти виконують 9 лабораторних робіт. Загальна кількість балів, яку отримують студенти за виконання та захист лабораторних робіт, становить 18 балів (максимально по 2 бали на кожну лабораторну роботу).

Шкала оцінювання знань студентів: національна та ECTS

<table>
<tr><td rowspan="2">Сума балів за всі види навчальної діяльності</td><td rowspan="2">Оцінка ECTS</td><td colspan="2">Оцінка за національною шкалою</td></tr>
<tr><td>Для іспиту, курсового проекту (роботи), практики</td><td>Для заліку</td></tr>
<tr><td>90–100</td><td>A</td><td>Відмінно</td><td rowspan="5">Зараховано</td></tr>
<tr><td>82–89</td><td>B</td><td rowspan="2">Добре</td></tr>
<tr><td>74–81</td><td>C</td></tr>
<tr><td>64–73</td><td>D</td><td rowspan="2">Задовільно</td></tr>
<tr><td>60–63</td><td>E</td></tr>
<tr><td>35–59</td><td>FX</td><td>Незадовільно з можливістю повторного складання</td><td>Не зараховано з можливістю повторного складання</td></tr>
<tr><td>0–34</td><td>F</td><td>Незадовільно з обов’язковим повторним вивченням навчальної дисципліни</td><td>Не зараховано з обов’язковим повторним вивченням навчальної дисципліни</td></tr>
</table>

СПИСОК ЛІТЕРАТУРИ

1. Острейковский В. А. Теория надежности : уч. пос. для ВУЗов / В. А. Острейковский. – М. : Высшая школа, 2003. – 463 с.

2. Диллон Б. Инженерные методы обеспечения надежности систем / Б. Диллон, Ч. Сингх. – М. : Мир, 1984. – 318 с.

3. Половко А. М. Основы теории надежности : практикум. / А. М. Половко, С. В.Гуров. – СПб : БХВ-Петербург, 2006. – 560 с.

4. Василевський О. М. Нормування показників надійності технічних засобів : навчальний посібник / О. М. Василевський, О. Г. Ігнатенко. – Вінниця : ВНТУ, 2013. – 160 с.

5. Васильків Н. М. Опорний конспект лекцій з дисципліни «Ефективність інформаційних систем» з освітньо-кваліфікаційного рівня «Спеціаліст» для спеціальності «Економічна кібернетика» / Н. М. Васильків. – Тернопіль : Економічна думка, 2005. – 98 с.

6. Шураков В. В. Надежность программного обеспечения систем обработки данных / В. В. Шураков. – М. : Финансы и статистика, 1987. – 272 с.

7. Карпова С. О. Методичні вказівки до виконання лабораторних робіт з дисципліни «Основи програмної інженерії» / С. О. Карпова. – Херсон : ХФ НУК, 2014. – 72 с.

8. Благодатских В. А. Стандартизация разработки программных средств : учеб. пособие / В. А. Благодатских, В. А. Волнин, К. Ф. Поскакалов; под ред. О. С. Разумова. – М. : Финансы и статистика, 2005. – 288 с.

9. Калбертсон Р. Быстрое тестирование / Р. Калбертсон, К. Браун, Г. Кобб. – М. : Изд. дом «Вильямс», 2002. – 374 с.

10. Столлингс В. Криптографическая защита сетей / В. Столлингс. – М. : Изд. дом «Вильямс», 2001.

11. Домарев В. В. Защита информации и безопасность компьютерных систем / В. В. Домарев. – К. : Диа-софт, 1999.

12. Інформаційна безпека інформаційно-комунікаційних систем.

Лабораторний практикум. Частина 1 – Комплекси засобів захисту інформації від НСД : навч. посіб. / М. В. Захарченко, В. Г. Кононович, В. Й. Кільдішев, Д. В. Голев; під ред. ак. МАІ М. В. Захарченка.– Одеса : ОНАЗ ім. О. С. Попова, 2011. – 168 с.

13. Інформаційна стійкість комп'ютерних технологій і мереж : навч. посіб. / А. В. Луговой, О. Г. Славко, П. П. Костенко, М. І. Гученко, М. М. Гузій. – Кременчук : Вид-во ПП Щербатих О. В., 2015. – 350 с.

Методичні вказівки щодо виконання лабораторних робіт з навчальної дисципліни «Інформаційна стійкість комп'ютерних технологій та мереж» для студентів денної форми навчання зі спеціальності 123 – «Комп'ютерна інженерія» освітнього ступеня «Магістр». Частина І

Укладачі: д. т. н., проф. М. І. Гученко,
к. т. н. П. П. Костенко,
асист. Н. Л. Сохін

Відповідальний за випуск зав. кафедри «Комп'ютерні та інформаційні системи» проф. А. В. Луговой

Підп. до др. _____. Формат 60×84 1/16. Папір тип. Друк ризографія.
Ум. друк. арк. ____. Наклад _____ прим. Зам. №_____. Безкоштовно.

Видавничий відділ
Кременчуцького національного університету
імені Михайла Остроградського
вул. Першотравнева, 20, м. Кременчук, 39600