Разрешающая способность датчиков 0,031ºС, период измерения и регистрации температуры 1сек, длительность непрерывной записи до 3 суток. Большая часть функций в измерителях выполняется программно, что дало возможность минимизировать объем специально разработанной аппаратной части. Управление режимами работы измерителей производится под управлением ПК, на экране которого в графическом и цифровом виде отображаются результаты измерений.

Для обеспечения необходимой точности измерений температуры разработана методика установки датчиков на поверхность трубопровода и калибровки датчиков в диапазоне измеряемых температур.

Практическое применение данного комплекса показало возможность определения величины теплопотерь с относительной погрешностью не более 3%. Использование нескольких комплексов позволяет определять тепловые потери в эксплуатационных режимах на участках с потребителя тепловой энергии.

1. Методические указания по определению тепловых потерь в водяных и паровых тепловых сетях МУ 34-70-080-84, М, 1985г .
2. Методические указания по определению тепловых потерь в водяных тепловых сетях РД 34.09.255-97, РАО энергетики и электрификации “ЕЭС России” .
3. *В.И.Мишустин, Ю.А.Чистяков* “Снижение тепловых потерь в тепловых сетях – одна из важнейших задач в общей проблеме энергосбережения” ФГУП “ВНИИМ им. Д.И. Менделеева”, Россия, 2000г.
4. Методика определения фактических потерь тепловой энергии через тепловую изоляцию трубопроводов водяных тепловых сетей систем централизованного теплоснабжения. М, “Издательство НЦ ЭНАС”, 2004г.
5. Інструкція з визначення теплових втрат через ізоляцію трубопроводів теплових мереж з використанням високоточної мікропроцесорної вимірювальної техніки И-2Т-2007, Київ 2007.

Поступила 30.08.2010р.

УДК 004.056.5

Б. Я. Корнієнко, к.т.н., НАУ, м. Київ, Г.О. Бойко, НАУ, м. Київ
О.С. Снігур, НАУ, м. Київ, Ю.О. Устимець, НАУ, м. Київ

ДОСЛІДЖЕННЯ МЕРЕЖНОГО РЕСУРСУ HONEY POT PLUS

This paper describes the basic principles of HoneyPot. A brief overview of the principles of operation and design of safety systems based on HoneyPot. Are reviewed, weaknesses and shortcomings of HoneyPot, identified the main drawback of HoneyPot. We present the modifications of HoneyPot using a plugin which solves the main disadvantage of HoneyPot – it’s passive.

Вступ

HoneyPot - це мережний ресурс, що є пасткою для зловмисника. Основне завдання HoneyPot - піддатися атаці, або несанкціонованому дослідженню, що дозволяє отримати модель зловмисника та визначити перелік засобів, які можуть нанести удари реально існуючим об'єктам безпеки. HoneyPot може бути реалізований за допомогою спеціально виділеного сервера, або простого мережного ресурсу, мета якого - привернути увагу зловмисників.

HoneyPot виконує збір необхідної інформації, після аналізу якої будується статистика методів, якими користуються зловмисники. Також визначається наявність нових рішень, які згодом використовуються для боротьби зі зловмисниками.

Існують два види HoneyPot:

- HoneyPot, які встановлені на виділеному сервері, що дозволяє максимально наблизити його до реального сервера, роль якого він виконує;

- емульований HoneyPot, що швидко поновлюється після атаки, а також чітко обмежується від основної операційної системи.

Для великих підприємств використовуються переважно HoneyPot, що встановлюються на виділеному сервері. Малі офісні мережі користуються емульованим HoneyPot.

HoneyPot широко застосовуються в Системах Виявлення Вторгнення (СВВ) як інструмент для створення правил безпеки. На основі аналізу активності в HoneyPot можливо створити правила або певний алгоритм дії, коли порушник намагається проникнути в систему. Наприклад, якщо після аналізу активності в HoneyPot було помічено, що невідомий хост з IP адресою 98.12.45.81 часто сканує порти системи, намагається встановити підключення, використовує заборонені методи для отримання доступу, то можливо встановити IP адресу 98.12.45.81 до чорного списку. Після чого СВВ будуть блокувати всі спроби підключення даного хоста.

Основна перевага HoneyPot- чітка модель порушника, що дає змогу отримати інформацію про причину і методи атаки на HoneyPot. Система не має авторизованої активності, тому всі дії спрямовані на систему - атака зловмисника.

Основним недоліком HoneyPot є пасивність системи, тобто системний адміністратор повинен власноруч переглядати події, що відбулися, аналізувати їх та на основі висновків, якщо вторгнення має місце, створювати нові правила та налаштування для системи безпеки.

Постановка задачі

Метою даної статті є дослідження основних принципів функціонування системи виявлення вторгнень та розробка системи HoneyPotPlus, що формує правила для виявлення небезпечного трафіку автоматично. Основою для системи була система з відкритим кодом – HoneyD.

Правила в системах виявлення вторгнення

Основна мета правил в системах виявлення вторгнення - це опис

характеристик атаки. В даний час не існує чітких стандартів що регламентують порядок створення правила.

Чітке правило повинно бути вузько направленим, щоб реагувати тільки на особливі характеристики нападу, одночасно воно повинно бути досить глибоко описаним, щоб охопити більшу кількість схожих атак.

Розроблена система підтримує правила для таких СВВ, як Bro [1] та Snort [2]. Bro має досить потужну мову програмування для правил, що дає змогу досить широко використовувати їх для опису руху трафіку та опису атак на різних рівнях. Snort на даний момент не є настільки потужним.

Шаблонно - рядковий алгоритм розпізнавання

Унікальність системи полягає у тому, що вона генерує правила. На відміну від мережених систем виявлення вторгнення, система не може взаємодіяти з базою даних до запуску, щоб порівнювати їх з активним трафіком.

Саме тому алгоритм шаблонної підстановки, що використовується в СВВ неможливо використовувати в даній системі. Замість того, система намагається розпізнавати структуру, яка вже була проаналізована раніше в HoneyPot: виділяємо певну частину з потоку трафіку і, використовуючи LCS алгоритм, намагаємося виявити схожості в пакетах корисного трафіку. Запропонована реалізація алгоритму LCS базується на дереві суфіксів, що використовуються для побудови різних рядкових алгоритмів. Використовуючи дерево суфіксів найдовший спільній підрядок двох рядків знаходиться за певний лінійний проміжок часу[7]. Пропонується декілька алгоритмів для побудови дерева суфіксів за лінійний проміжок часу [8]. Запропоновано використовувати алгоритм Укконена.

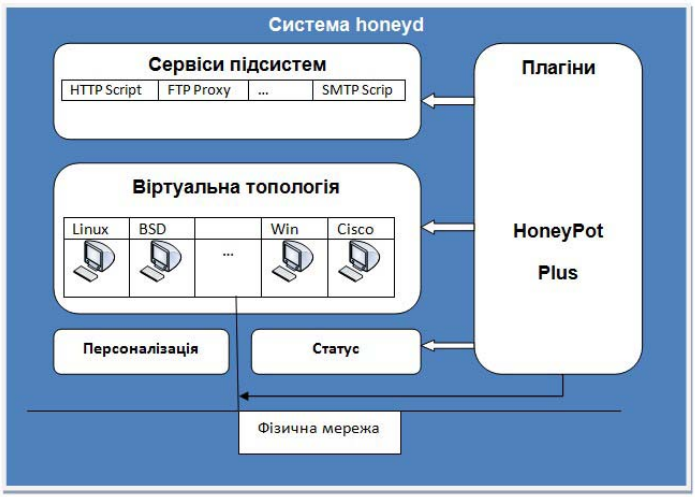

Рис. 1. Структурна схема HoneyD

Архітектура HoneyPotPlus

В процесі розробки системи змінено концепцію HoneyD і додано дві нові можливості: плагіну структури та зворотній зв'язок подій. Плагін структура дозволяє створювати розширення, що логічно відрізняються від кодової бази HoneyD, в той час як зворотній зв'язок дає можливість інтегрувати плагіни до активного середовища HoneyPot. На даному етапі зворотній зв'язок інформує плагіни про обмін пакетами, про статус з'єднання, та про рух даних (відправка або прийом). Система HoneyPotPlus впроваджується як плагін для HoneyD.

Інтеграція розробленої системи до HoneyD має ряд переваг, ніж використання її як самостійної системи:

- Розроблена система вимагає доступ до мережного трафіку. Якщо система використовується самостійно, хорошим вибором для вирішення цього питання можу бути libpcap [3]. В HoneyD вже використовується libpcap для моніторингу трафіку та симуляції системи.
- Уникнення холодного запуску системи: найбільша перевага полягає у тому, що HoneyD не просто пропускає трафік, аналізуючи його, а створює відповіді на запити за допомогою імітації стеку і конфігурації системи. Інтеграцією HoneyPotPlus до HoneyD уникаємо десинхронізації існуючого з'єднання: коли HoneyD отримує пакет, він встановлює нове з'єднання, при цьому HoneyPotPlus проінформована, що починається з'єднання. Тому питання: чи розпізнає система початок з'єднання, втрачає актуальність, порівняно з самостійними системами [4-6].

Алгоритм створення правил

Підхід полягає в тому, щоб створити систему, що не має ніяких знань про рівень протоколу і кожний отриманий пакет розглядається по одному і тому ж алгоритму:

- якщо для нового пакету є характеристика з'єднання - то така характеристика поновлюється, якщо ж не існує - то вона створюється;
- якщо пакет має вихідний статус - обробка зупиняється;
- HoneyPotPlus дає змогу аналізувати пакети на мережному та транспортному рівнях;
- для кожного з'єднання, що зберігається HoneyPotPlus виконує порівняння заголовків для знаходження IP мереж, послідовності номерів TCP;
- якщо з'єднання мають однакові порти призначення, HoneyPotPlus намагається проаналізувати їх методом шаблону пакетів, якими обмінюються;
- якщо не було створено жодного корисного підпису, обробка пакетів зупиняється;
- періодично, база даних з підписами копіюється на жорсткий диск.

Відстеження з'єднань

HoneyPotPlus утримує певну кількість TCP, UDP з'єднань, але потребує

дуже специфічних умов для їх утримання. Оскільки, головна задача – генерація правил шляхом порівняння нового трафіку в HoneyPot з попередньо вивченим, тому не маємо змоги відкинути статус з'єднання одразу після розірвання з'єднання. Замість цього, маркуємо з'єднання, як розірване і намагаємося утримувати його якомога довше, чи до того моменту, коли буде ясно, що не отримаємо ніякої користі від його зберігання.

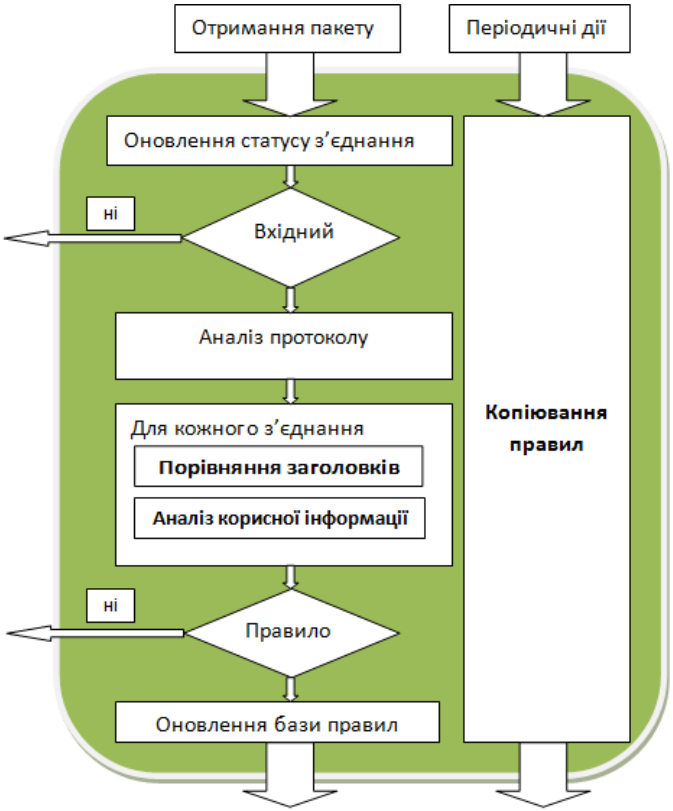

Рис. 2. Алгоритм створення правил

З'єднання, через які проходить більша кількість інформації - потенційно важливі, оскільки саме в них маємо змогу знайти співпадання трафіку. Система повинна запобігати агресивному сканування портів, оскільки це призводить до надлишку в хеш-таблиці, що, в свою чергу, може призвести до втрати цінних з'єднань. Саме тому, всі TCP, UDP з'єднання зберігаються в двох стадіях: спочатку з'єднання зберігається в таблиці «ручного розподілу» і потім переміщається до таблиці «встановлено», коли відбувається обмін корисними даними.

Система виконує переформування потоку даних: для з'єднання TCP переформовуємо потік до останнього переданого байту. Переформований потік даних являє собою список всіх повідомлень, де повідомлення представляється у вигляді корисної інформації, що була передана в одному напрямку без корисної інформації, що була передана в іншому напрямку. Наприклад, для звичайного HTTP запит зберігається у вигляді двох повідомлень: одне - для HTTP запиту, інше - для HTTP відповіді. Для UDP також створюємо повідомлення, що містить всю корисну інформацію в одному напрямку без зворотної інформації (рис.3).

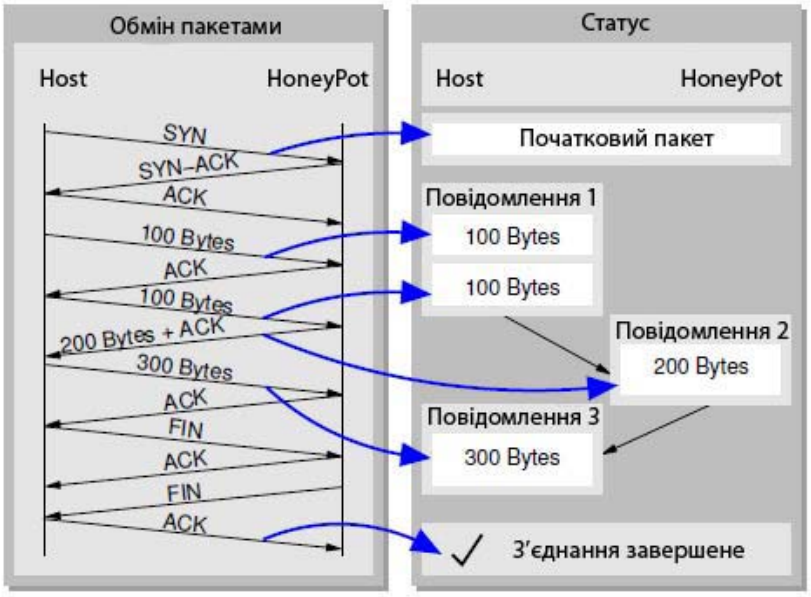

Рис. 3. Відстеження з'єднань

Аналіз протоколу

Після того, як статус з'єднання було оновлено, HoneyPotPlus створює порожнє правило для потоку даних та розпочинає перевірку пакетів. Кожне правило має унікальний ідентифікатор та зберігає знайдені особливості вивченого трафіку незалежно від конкретної МСВВ(Мережної системи виявлення вторгнення). Запис правила поступово розширюється, в залежності від того, скільки було виявлено особливостей.

В даний момент виконуємо аналіз протоколу на мережному та транспортному рівнях для IP, TCP та UDP заголовків пакету, використовуючи техніку header-walking, що раніше застосовувалася, як метод нормалізації трафіку.

Після цього HoneyPotPlus виконує порівняння заголовків з кожним

з’єднанням однакового типу (TCP або UDP), яке зберігається в базі. Якщо з’єднання, що зберігається в базі вже було переміщено в другий рівень хеш-таблиці, HoneyPotPlus намагається знайти відповідні йому повідомлення і при цьому використовує заголовки, які відносяться до цього повідомлення.

Якщо виявляються співпадання (наприклад, співпадання IP ідентифікатора або діапазона мережі), аналізоване правило клонується та стає відповідним для даного потоку даних. Знайдені особливості потім записуються в нове правило.

Знаходження зразка в потоці даних

Після аналізу протоколу, HoneyPotPlus розпочинає аналіз переформованого потоку даних .

Горизонтальне розпізнавання: нехай кількість повідомлень в з’єднанні після оновлення статусу становить – n. HoneyPotPlus намагається провести порівняння за шаблоном на n повідомлення на всіх збережених з’єднаннях з однаковим портом призначення в HoneyPot, використовуючи LCS алгоритм, щоб взаємодіяти з рядками напряму (рис. 4).

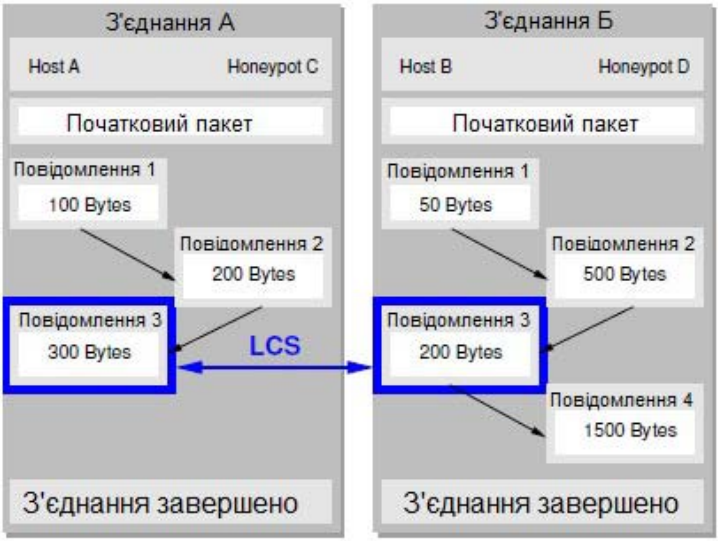

Рис. 4. Горизонтальне розпізнавання

Вертикальне розпізнавання: Одночасно HoneyPotPlus виконує конкатенацію всіх вхідних повідомлень з одного індивідуального з’єднання, після чого передає два конкатенованих повідомлень з різних з’єднань до LCS алгоритму. Суть полягає в тому, що горизонтальне розпізнавання не зможе розпізнати схожість в інтерактивній сесії, наприклад Telnet, в той час, як вертикальне розпізнавання буде працювати. Але головне те, що вертикальне розпізнавання визначає динаміку TCP: конкатенація створює ефект розділення всього потоку даних в окремі повідомлення (рис.5).

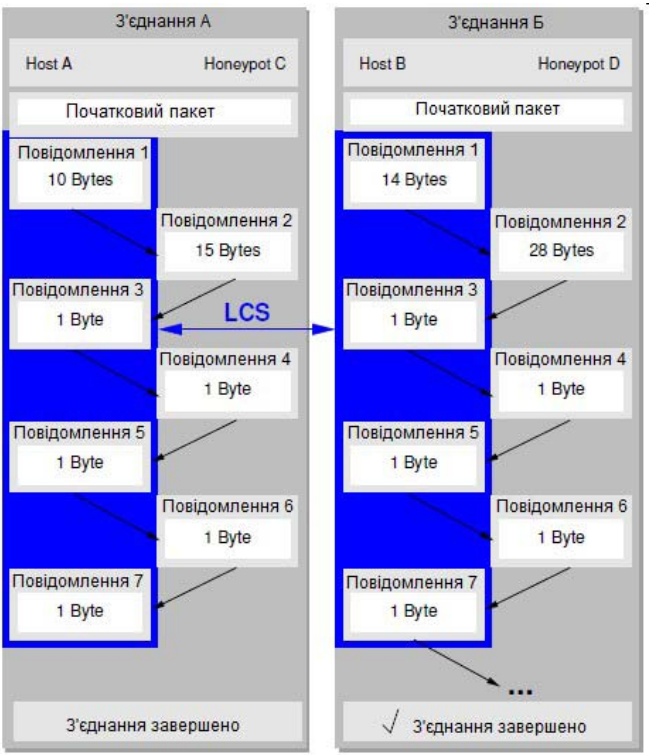

Рис. 5. Вертикальне розпізнавання

Цикл існування правила

Якщо правило не містить ніяких фактів, то на цьому етапі обробка пакетів припиняється. В іншому випадку, HoneyPotPlus перевіряє, як можна використати правило для покращення бази правил, що являє собою історію створених правил.

База правил реалізована у вигляді послідовності, що обмежена максимальним розміром. Одразу після того, як створене нове правило для зберігання в базі, старе видаляється. Видалені правила не втрачаються, оскільки дані з бази правил дублюються через певний інтервал часу.

HoneyPotPlus намагається зменшити кількість правил ,що дублюються, використовуючи алгоритм накопичення. З цією метою визначили кількість реляційних операцій для генерації правил:

- sig1=sig2: правила ідентичні. Цей оператор оцінюється як «так», коли sig1 та sig2 співпадають у всіх атрибутах крім тих, що можливо виразити у вигляді списку в кінцевому правилі (наприклад, порт призначення);
- Sig 1 U sig2: правило 1 визначає лише частину особливостей правила 2. Ця частина включає будь-які схожості, що знайдені за допомогою LCS

алгоритма: послідовність байтів першого правила слабша, ніж другого, в той час перше правило є частиною другого.

Якщо нове правило являє собою суперпозицію уже існуючого, то воно модифікує старе правило або, в іншому випадку, просто додається до бази.

Рис. 6. Спектр трафіку

Вихідне правило

Вміст бази правил періодично передається на вихідний модуль, який виконує збереження правил. На даний момент існують реалізації модулів, що конвертують правила в Bro або псевдо-Snort формат, та модулі, що виконують запис правил до файлу.

Реалізація системи

Протягом 24 годин отримано 224 Кб інформації, 557 TCP з'єднань, 145 UDP з'єднань та 27 ICMP пінчів (рис.6).

HoneyPotPlus створив 38 правил для хостів, що сканували порти. 25 правил було створено на основі частин даних з трафіку. Найдовшу строку було сформовано для інтернет - черв'яка: HoneyPotPlus зміг створити правило для Slammer та CodeRED II Worm. Правило було створено лише після 6 запитів Slammer та 3 CodeRed II. Також HoneyPotPlus не повідомляв про довгі запити HTTP GET від CodeRed II.

Висновки

Розроблена система HoneyPotPlus, що створює правила для МСВВ шляхом аналізу трафіку в HoneyPot. Система створює якісні правила для підключень, що використовують звичайні користувачі. Також система досить добре створює правила для інтернет - черв'яків.

1. *V. Paxson.* Bro: A System for Detecting Network Intruders inReal-Time,. *Computer Networks (Amsterdam, Netherlands: 1999)*, vol. 31, no. 23-24, pp. 2435-.2463.
2. *M. Roesch.* Snort: Lightweight Intrusion Detection for Networks,. In *Proceedings of the 13th Conference on Systems Administration*, 1999,pp. 229-238.
3. *C. Stoll*. *The Cuckoo's Egg*. Addison-Wesley, 1986.
4. *W. R. Cheswick*. An Evening with Berferd, in which a Cracker is lured, endured, and studied,. in *Proceedings of the 1992 Winter USENIX Conference*, 1992.
5. *L. Spitzner. Honeypots: Tracking Hackers*. Addison-Wesley, 2003.
6. *N. Provos*. HoneyD - A Virtual Honeypot Daemon,. in *10th DFN-CERTWorkshop,*

Hamburg, Germany, February 2003.
7. *D. Guseld. Algorithms on Strings, Trees and Sequences*. Cambridge University Press, 1997.
8. *P. Weiner*. Linear pattern matching algorithms,. in *Proceedings of the14th IEEE Symposium on Switching and Automata Theory*, 1973, pp.1.11.

Поступила 1.10.2010р.

УДК 681.3

С. М. Головань, А. М. Давиденко, Л. М. Щербак

ПРОЦЕСИ ЖИТТЄВОГО ЦИКЛУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В СИСТЕМАХ ЕЛЕКТРОННОГО ДОКУМЕНТООБІГУ

In this work the processes of software life cycle - the process of production problems, the process of development, improvements and services and the purchase and distribution of software.

Вступ

На сьогодні організації прагнуть використати потужні можливості сучасних інформаційних систем, їх високу швидкість дії та програмного продукту необхідного для створення, відправлення, передавання, одержання, зберігання, оброблення, використання та знищення різної інформації електронного документообігу. Системи, які використовуються в цих операціях, можуть бути класифіковані як системи автоматизованого виробництва, складовою компонентою якого є програмне забезпечення.

Для досягнення ефективності програмного продукту необхідно впорядкувати різні цілі, стратегії і методики захисту інформації з обмеженим доступом для кожного організаційного рівня, а також зменшити вартість системи за рахунок оптимізації програмного забезпечення.

В даній роботі розглянуто загальний підхід до процесу життєвого циклу програмного забезпечення, яке включає програмні продукти з захисту інформації.

Процеси життєвого цикл програмного забезпечення

Процеси життєвого цикл програмного забезпечення – процес постановки проблеми і процес її розробки, а також удосконалення (доповнення, модернізація, а при необхідності розробка нового програмного забезпечення) та послуги (рис. 1).

Процес постановки проблеми починається з вимог споживача та закінчується повним описом алгоритму роботи програмного забезпечення. Процес розробки починається з технічних вимог і закінчується постановкою системи автоматизованого виробництва (електронних документів та

