9. *Vaudenay S.* On the security of CS-cipher // Fast Software Encryption. – FSE'99, Proceedings. – Springer Verlag, 1999. – P. 260 – 274.

Поступила 22.03.2006

УДК 621.372

Корченко О. Г., Ануфрієнко К.П.

КЛАСИФІКАЦІЯ УРАЗЛИВОСТЕЙ В ПОЧАТКОВИХ КОДАХ

Наприкінці минулого століття на ринку програмних засобів забезпечення безпеки відбувся значний сплеск: з'явились та набули поширення різноманітні антивірусні програми, міжмережеві екрани, криптографічні та інші програмні засоби. Проте після багатьох років використання цих засобів кількість атак на ресурси комп'ютерних систем (КС) продовжує зростати [1]. Інциденти з безпекою на прикладному рівні взаємодії комп'ютерних систем посідають чільне місце серед найбільш значних проблем безпеки інформаційних технологій. Традиційні методи та засоби захисту інформаційних технологій в основному зосереджені на мережевому й периметровому захисті, лишаючи поза увагою проблему захищеності програмного забезпечення (ПЗ). В результаті ПЗ виявляється зазвичай слабкою ланкою в системі захисту. Так, за даними Національного інституту стандартів і технологій США, 93% уразливостей, які знаходять, – це уразливості в ПЗ [2].

У зв'язку зі специфікою розробки ПЗ, уразливості у ньому виникають в основному в процесі розробки під час написання початкового коду. Кількість уразливостей в початкових кодах невпинно зростає. Згідно з даними CERT Coordination Center за 1998 р. було зафіксовано 262 уразливості, тоді як за 2002 р. – 4131 уразливість [3].

І в той час, як більшість розроблювачів оцінюють функціональні можливості, продуктивність і здатність до інтеграції додатків, відсутність перевірки їх захищеності під час процесу розробки може мати серйозні наслідки. Помилка в коді програми може призвести до таких катастрофічних збитків, як втрата інтелектуальної власності, коштів або важливих даних.

Проблема аналізу захищеності ПЗ, вибору ефективних методів і засобів розробки захищеного ПЗ (ЗПЗ) та розробки систем виявлення уразливостей в початкових кодах значною мірою залежить від організації життєвого циклу зазначеного забезпечення, мови програмування, конкретних реалізацій алгоритмів обробки вхідних даних, можливостей порушень характеристик безпеки та інших чинників. Ефективність її розв'язання в першу чергу пов'язана із визначенням того, на які класи уразливостей розраховані ті чи інші методи та засоби їх виявлення.

Проблема уразливостей і їх виявлення досліджується досить давно. У роботах [1-9] різними авторами розглядається використання уразливостей в початкових кодах для здійснення атак, а також методи та засоби виявлення цих уразливостей чи запобігання їх появі. Наведені у цих роботах декілька варіантів класифікацій уразливостей не охоплюють широкого спектра ознак і не характеризуються системним узагальнюючим підходом, що можна було б використати при розв'язанні зазначеної проблеми. Крім того, деякі класифікації дещо звужені, наприклад, до таких об'єктів, як операційні системи (класифікація уразливостей Т. Аслама (T. Aslam) в операційній системі Unix [4]). Найновіший структурований підхід до класифікації уразливостей [5] полягає у тому, що уразливості як об'єкти мають властивості (початковий код, компоненти ПЗ, версію програми, проломи, властивості експлойту та ін.) – атрибути (некоректна довжина, вплив, наслідки, розташування коду експлойту тощо) із певним значенням. Класифікація здійснюється шляхом призначення уразливості набору атрибутів із конкретними значеннями. В результаті різні уразливості мають відмінний набір ознак. Такий підхід нагадує більше

спеціалізовану структуровану мову опису і не відповідає критеріям системності та узагальненості. Отже, недосконалий рівень абстракції є серйозною проблемою для відомих класифікацій. Значна частина розглянутих класифікацій стикалася з тим, що деякі уразливості можна віднести одразу до декількох типів чи категорій, що суперечило поставленій задачі. Запропонований в даній статті підхід вирішує ці проблеми.

На основі проведеного дослідження відомих класифікацій та методів і засобів експлуатації уразливостей в початкових кодах пропонується узагальнена класифікація, яка може бути застосована для аналізу різноманітного ПЗ, представленого у початкових кодах. Спектр уразливостей досить різнорідний, тому основним принципом, за яким можна найбільш ефективно (відповідно до поставленої проблеми) здійснити класифікацію, буде ознаковий.

Для того, щоб запобігти неоднозначності в тлумаченнях використовуваних далі термінів, на підґрунті [10] та [11] дамо низку визначень.

Комп'ютерна система (*computer system*) – компоненти, які функціонують як єдине ціле і призначені для розв'язання визначеного класу задач [10].

Компоненти (*components*) комп'ютерної системи (ККС) – взаємозв'язана сукупність апаратних засобів, відповідного ПЗ, фізичних носіїв інформації та даних (іноді навіть обслуговуючого персоналу) [10].

Ресурс (*resource*) – будь-який з ККС, а також сама КС і надані нею можливості, наприклад, спільного використання мережевого каталогу чи принтера [10].

Конфіденційність ресурсів (*resource confidentiality*) – характеристика безпеки ресурсів, що відображає їх властивість невиявленості й доступності без відповідних повноважень. Фактично ресурси не можуть бути доступні або розкриті неавторизованій стороні, тобто для неї їх нібито немає. В свою чергу, авторська сторона (наприклад, обслуговуючий персонал, користувачі, програми та ін.), якій надано відповідні повноваження, має повний доступ до ресурсів [12].

Цілісність ресурсів (*resource integrity*) – характеристика безпеки ресурсів, що відображає їх властивість протистояти несанкціонованій зміні. Наприклад, користувач, що накопичує інформацію, має право сподіватися, що вміст його файлів залишиться незмінним, незважаючи на цілеспрямовані впливи, відмови програмних або апаратних засобів. За цією характеристикою ресурси не зазнають будь-якої зміни з боку неавторизованої сторони [12].

Доступність ресурсів (*resource availability*) – характеристика безпеки ресурсів, що відображає їх властивість, яка полягає в можливості їх використання у заданий момент часу відповідно до пред'явлених повноважень. Фактично авторська сторона, за потреби, відразу, в будь-який момент часу отримує необмежений доступ до необхідного ресурсу [12].

Атака (*attack*) – заходи, які вживаються для підриву безпеки КС (відмова в обслуговуванні, порушення характеристик безпеки і та ін.) [12].

Програма (*program*) – упорядкована послідовність команд, що підлягає обробці [10].

Початковий код (*source code*), або початковий текст (програми), – текст програми алгоритмічною мовою програмування перед процедурою трансляції або інтерпретації [10].

Програмне забезпечення (*software*) – загальне поняття, що описує програми для КС на відміну від апаратних ККС, тобто це програма або сукупність програм, що взаємопов'язані з метою розв'язання певних задач. При цьому неважливо, в якому вигляді представлені програми: в початкових текстах або у виконуваному коді [11].

Захищене програмне забезпечення (*secure software*) – це ПЗ, яке забезпечує конфіденційність, цілісність та доступність інформації, а також цілісність та доступність інших ресурсів КС [6].

Уразливість (*vulnerability*) – слабке місце в системі (її неможливість протистояти загрозам), яке може бути використане для отримання несанкціонованого доступу (НСД) [10].

Пролом, або пролом у захисті (*security flaw*), – недолік, помилка чи дефект у ПЗ або його компоненті, що, за наявності певних умов, може призвести до уразливості у ПЗ [5].

Уразливість у програмному забезпеченні (*software vulnerability*) – це пролом у захисті за наявності певних умов, тобто такий недолік ПЗ, при якому навіть правильне використання програм не попередить успішних атак, наприклад, отримання привілеїв у системі, втручання в її роботу, псування даних або неправомочне отримання прав у системі [5, 6].

Уразливість в початкових кодах (*source code vulnerability*) – уразливість у ПЗ, представленому в початкових кодах, причини якої криються у навмисних чи ненавмисних несинтаксичних помилках під час реалізації мовою програмування алгоритмів, функцій (процедур, модулів), класів та об'єктів, а не в помилках у програмній архітектурі. Така уразливість не могла виникнути під час процедури трансляції або компоновки і не пов'язана безпосередньо з налагоджуванням, конфігуруванням, функціональними можливостями або особливостями інтерфейсу, ергономіки та дизайну програм.

Експлойт (*exploit*) – програмний засіб, який використовує можливості, надані уразливістю в коді програми для руйнування її захищеності.

Експлуатація уразливостей (*vulnerability exploitation*) – використання уразливостей в початкових кодах за допомогою спеціальних програмних засобів та експлойтів для здійснення атак.

Шкідливий код (*harmful code*) – спеціально сформований код, що передається уразливій програмі, замаскований під дані. Може виконуватись певна частина цього коду або перезаписуються інші дані у пам'яті. Шкідливий код може бути як у формі початкового тексту, так і у формі так званого байт-коду (*byte code*). Байт-код є майстерно написаним на асемблері кодом, що є закінченою програмою і не містить деяких спеціальних символів [7]. Найпоширенішим прикладом байт-коду є шелл-код.

Шелл-код, або код оболонки (*shellcode*), – це фрагмент машинного коду, що в результаті виконання передає управління командній оболонці (консолі, *англ. shell*; наприклад, Unix shell, command.com у MS-DOS та cmd.exe в операційній системі Microsoft Windows XP). Шелл-код може бути використаний як корисне навантаження експлойту, забезпечуючи зломнику доступ до командної оболонки у КС [7].

Виходячи з визначення уразливостей в початкових кодах пропонується класифікувати їх за наступними базовими ознаками (що в основному характеризують уразливості в контексті експлуатаційних умов): походження задіяного в експлуатації виконуваного коду; спроможність до автоматизованої експлуатації; дистанційність взаємодії з початковим кодом; ступінь організації; механізм експлуатації; порушення характеристик безпеки; тип базового ресурсу; тип ініціюючого ресурсу; намір. Уразливості не класифікуються за такими ознаками, як, наприклад, ступінь ризику, наслідки, направленість результату використання, складність виявлення, складність усунення та іншими тому, що ці ознаки не характеризують саме уразливість, а характеризують наслідки її експлуатації, які залежать від зовнішніх по відношенню до уразливості факторів (виду експлойту, типу атаки, намірів хакеру тощо).

Розглянемо класифікацію уразливостей в початкових кодах за вищевизначеними ознаками докладніше.

За походженням задіяного в експлуатації виконуваного коду уразливості бувають код-екзогенні (*code exogenic*), код-ендогенні (*code endogenic*) та код-полігенні (*code polygenic*). Найпоширенішими є код-екзогенні уразливості. Такі уразливості дозволяють програмі сприйняти спеціально сформовані дані (передані зловмисником), а потім виконати шкідливий код, що міститься у них. Код-ендогенні уразливості полягають в потенційній експлуатації існуючого коду програми у зловмисних цілях. Наприклад, наприкінці 2005 р. була виявлена проблема з обробкою зображень у форматі WMF. Це була не типова помилка на зразок переповнення буферу. Формат WMF створювався у час, коли обробка зображень на ПК відбувалася повільно, тому у ньому передбачена можливість перервати виконання коду. Виявлені уразливості дозволяють спровокувати аварійне завершення роботи програми (використовуючи існуючий код програми), тобто викликають відмову в обслуговуванні. Якщо уразливості можуть проявляти і код-екзогенні, і код-ендогенні властивості, то такі уразливості називаються код-полігенними.

За спроможністю до автоматизованої експлуатації уразливості можна поділити на автоматизовано спроможні (*automated exploitable*) та автоматично спроможні (*automatic exploitable*). Автоматизовано спроможні експлуатуються за постійною участю оператора з використанням широкого спектру програмних засобів (експлойтів, програмних закладок, шпигунських та різноманітних шкідливих програм). Експлуатація автоматично спроможних уразливостей реалізується без участі людини і, як правило, з використанням спеціалізованих програмних засобів, функціонування яких базується на вірусних технологіях.

За дистанційністю взаємодії з початковим кодом (зловмисника, експлойту, ініціюючого ресурсу) уразливості поділяються на локалізовані (*localized, locally exploitable*) та дистанційовані (*remotely exploitable*). Дистанційовані уразливості можуть використовуватися через мережу без будь-якого переважного доступу до вразливої КС. Локалізовані уразливості потребують переважного доступу до вразливої КС і зазвичай надають можливість підвищити привілеї для користувача, що запускає експлойт, вище того рівня, який мав користувач.

За ступенем організації уразливості можна поділити на елементарні (*elementary*), складні (*compound*) та системні (*systemic*). Елементарні уразливості є нескладними в реалізації, пов'язані з порушенням граничних умов при використанні масивів, помилковим використанням операторів, помилками в окремих процедурах чи функціях, помилковим вживанням типів даних, відсутністю перевірки вхідних даних та інше, наприклад, заниження або завищення на одиницю розміру масиву чи індексу елемента масиву, присвоєння змінній даних неправильного типу, передача функції рядку зі спеціальним символом форматування тощо. Складні уразливості є комбінацією елементарних, що пов'язані або залежать одні від одних. Виявляти складні уразливості досить важко. Системні уразливості будуються навмисно на основі певного системного підходу з використанням елементарних уразливостей для ефективної експлуатації з метою реалізації певних функцій, наприклад, обхід захисних механізмів, проникнення до КС, отримання прав доступу, несанкціоноване копіювання даних, стеження тощо.

За механізмом експлуатації уразливості бувають такі: ін'єкції коду (*code injection*); переповнення буферу (*buffer overflow, buffer overrun*); рядку форматування (*format string*); приведення до канонічного вигляду (*canonicalization*). Цей перелік не є вичерпним і може бути розширений.

Уразливості ін'єкції коду – уразливості, що дозволяють здійснити ін'єкцію стороннього шкідливого коду (сформованого зловмисником) у запит, що передається від однієї програми до ККС. В свою чергу вони поділяються на уразливості ін'єкції коду сценаріїв (*Script Injection*) та мов розмітки (*Markup Language Injection, ML Injection*), SQL-коду (*SQL Injection*), LDAP-коду (*LDAP Injection*), а також уразливості міжсайтового виконання сценаріїв (*Cross-site Scripting, XSS*) та ін. Регулярно з'являються нові різновиди уразливостей ін'єкції коду.

Ін'єкція SQL-коду – можливість модифікувати запит до SQL-сервера, що відправляється програмою. Проблема ін'єкції SQL-коду вперше була описана в статті Рейна Фореста Паппі (Rain Forest Puppy) "NT Web Technology Vulnerabilities" у журналі Phrack Magazine у 1998 р. [8]. Мова запитів Structured Query Language (*SQL*) є спеціалізованою мовою програмування, що дозволяє створювати запити до серверів СУБД. Більшість серверів підтримує цю мову у варіантах, стандартизованих ISO і ANSI. У більшості сучасних СУБД присутні розширення діалекту SQL, специфічні для даної реалізації (T-SQL в Microsoft SQL Server, PL SQL в Oracle і т. д.). Багато програм використовують дані, передані користувачем, для створення динамічних Web-сторінок. Якщо інформація, отримана від клієнта, належним чином не верифікується, можлива модифікація запиту до SQL-серверу, що відправляється програмою. Запит буде виконуватися з тим же рівнем привілеїв, з яким працює компонент програми, що виконує запит (сервер СУБД, Web-сервер і т. д). У результаті зловмисник може одержати повний контроль над сервером СУБД і навіть його операційною системою.

Спрощений протокол доступу до служби каталогу (Lightweight Directory Access Protocol, LDAP) – відкритий протокол для створення запитів і керування службами каталогу, сумісними зі стандартом X.500. Протокол LDAP працює поверх транспортних протоколів Internet (TCP/UDP). Ін'єкція LDAP-коду – можливість модифікувати LDAP-запит. Програма може використати дані, надані користувачем для створення запитів за протоколом LDAP при генерації динамічних Web-сторінок. Якщо інформація, отримана від клієнта, належним чином не верифікується, є можливість модифікувати LDAP-запит. Запит буде виконуватися з тим же рівнем привілеїв, з яким працює компонент програми, що виконує запит (сервер СУБД, Web-сервер і т. д.). Якщо даний компонент має право на читання або модифікацію даних у структурі каталогу, зловмисник одержує ті ж можливості.

Міжсайтове виконання сценаріїв дозволяє передати серверу виконуваний код, що буде переспрямований браузеру користувача. Цей код звичайно створюється мовами HTML, JavaScript, але можуть бути використані VBScript, ActiveX, Java, Flash або інші,підтримувані браузером технології. Переданий код виконується в контексті безпеки (або зоні безпеки) уразливого сервера. Використовуючи ці привілеї, код одержує можливість читати, модифікувати або передавати важливі дані, доступні за допомогою браузера.

Уразливості ін'єкції коду сценаріїв залежать від специфіки сценаріїв та поділяються відповідно на такі: серверних розширень (*SSI Injection*), сценаріїв JavaScript, ASP, PHP, Perl та ін. Усі види ін'єкції коду сценаріїв мають однаковий механізм: зловмисник передає спеціально сформований код Web-сервера, ПЗ якого виконає цей код.

Розглянемо докладніше ін'єкцію серверних розширень, що дозволяє зловмисникові передати виконуваний код, який надалі буде виконаний на Web-сервері. Така уразливість звичайно полягає у відсутності перевірки даних, наданих користувачем, перед збереженням їх в інтерпретованому сервером файлі. Перед генерацією HTML-сторінки сервер може виконувати сценарії, наприклад, Server-site Includes (*SSI*). У деяких ситуаціях вихідний код сторінок генерується на основі даних, наданих користувачем. Якщо передати серверу оператори SSI, він може одержати можливість виконання команд операційної системи або включити в неї заборонений зміст при наступному відображенні.

Уразливості ін'єкції мов розмітки мають загальний для ін'єкції коду механізм експлуатації, але відрізняються специфікою технології мов розмітки, тому бувають відповідно такі: ін'єкція HTML-коду (*HTML Injection*), ін'єкція XPath-коду (*XPath Injection*) та ін. Так, якщо запити XPath генеруються під час виконання на основі користувацького введення, є можливість модифікувати запит з метою обходу логіки роботи програми. Уразливість, що дозволяє це зробити, називається ін'єкцією XPath-коду. Мова XPath розроблена для надання можливості звертання до частин документа мовою XML. Синтаксис XPath близький до мови SQL запитів. У мові XPath XML-документ розглядається у вигляді дерева, в якому кожна XML-конструкція подається окремим вузлом.

Переповнення буфера – уразливість, що дозволяє програмі записувати дані за межами виділеного в пам'яті буфера. Переповнення буфера звичайно виникає внаслідок помилки й невірного використання таких мов, як C або C++, які не є "мовами із захистом пам'яті". Один із результатів переповнення полягає в тому, що правильні дані можуть бути спотворені. Переповнення буфера також є найпоширенішим способом злому ОС, тому що керуючі дані програми часто знаходяться в областях пам'яті, розташованих поруч із буферами даних. У результаті переповнення буфера комп'ютер може змінити значення інших даних у пам'яті або виконати довільний потенційно шкідливий код, що був переданий уразливій програмі як дані.

Уразливість переповнення буфера буває декількох видів залежно від сегмента пам'яті програми, що переповнюється. Взагалі пам'ять програми поділяється на п'ять сегментів: текст (*text*), дані (*data*), bss (*bulk storage system* – пристрій масової пам'яті), купа (*heap*) та стек (*stack*). Наразі відомі такі види переповнення буфера: переповнення стека (*stack overflow, stack smashing*), переповнення купи та bss (*heap and bss overflow*) [7]. У сегменті пам'яті bss записуються неініціалізовані глобальні та статичні змінні програми, цей сегмент

пам'яті доступний для запису, але розмір цього сегмента фіксований. Сегмент купи відводиться для інших змінних програми. Розмір купи не фіксований: вона може зменшуватися або збільшуватися у міру необхідності. Сегмент стека також має змінний розмір і використовується як тимчасова пам'ять для збереження контексту під час виклику функцій.

Уразливість переповнення буфера у вигляді переповнення стека виникає, коли буфер (виділений у стека) перезаписується даними, об'єм яких перевищує його розмір. Розміщені у стеку змінні фізично розташовані поряд з адресою повернення для коду, що викликав функцію. У результаті справжня адреса повернення перезаписується підставною адресою [7].

У випадку уразливостей переповнення в сегментах купи та bss не має адреси повернення, яку треба змінити. Ці види переповнення залежать від важливих змінних, що зберігаються у пам'яті слідом за буфером, який можна переповнити. Значення такої змінної, що містить, наприклад, права доступу користувачів або результат авторизації, можна змінити і надати повні права доступу або аутентифікації у системі. А якщо за буфером, уразливим для переповнення, зберігається покажчик на функцію, то, змінивши цей покажчик, можна примусити програму звернутися за іншою адресою пам'яті, коли відбудеться звернення до цього покажчика функції. Оскільки уразливості переповнення у сегментах купи та bss набагато більше залежать від структури пам'яті у програмі, ці види уразливостей значно важче виявити [7].

Уразливість рядку форматування – уразливість, яка пов'язана з неграмотним застосуванням функцій форматного виводу (таких як printf(), fprintf(), sprintf() та ін. мовою С), що дозволяє зловмиснику, використовуючи спеціально сформований рядок форматування, змінити шлях виконання програми у пам'яті. Клас цих уразливостей виник нещодавно. Якщо атакуючий передає програмі рядок, що містить символи форматування ("%f", "%p", "%n" і т. д.), то в нього з'являється можливість: виконати довільний код на сервері; зчитувати значення зі стека; викликати помилки в програмі або відмову в обслуговуванні.

Досить часто рядок форматування відносять до переповнення буфера. Незважаючи на те, що ці класи уразливостей використовуються з однією метою (наприклад, для передачі управління шелл-коду, що надає можливість зловмиснику використовувати командну оболонку), по суті вони є відмінними. Так, проблема рядку форматування пов'язана з передаванням функції форматного виводу неперевіреного аргументу, а переповнення буфера – із помилкою меж (*boundary error*) масивів даних (або рядків). Рядок форматування використовується для запису специфічних значень до певної області пам'яті, а переповнення буфера не дає можливості вибрати, яка конкретна область пам'яті буде перезаписана.

Приведення до канонічного вигляду – це перетворення різноманітних еквівалентних форм імені ресурсу (наприклад, імені файлу, імені каталогу) до єдиного, стандартного вигляду – канонічного. Уразливості приведення до канонічного вигляду полягають у тому, що програма може зробити невірний висновок на основі неканонічного подання імені. Наприклад, відомо, що операційні системи сімейства Windows успадкували для зворотної сумісності деякі імена файлів у MS-DOS. Насправді це не файли, а пристрої, такі як послідовний порт (*aux*) та принтер (*lpt1* і *prn*). Використовуючи цей пролом, хакери отримали можливість примусити Windows 95 та Windows 98 звертатися до цих пристроїв. Коли Windows намагалася проінтерпретувати ім'я пристрою як файловий ресурс, відбувалось недопустиме звернення до ресурсу, що звичайно закінчувалося відмовою в обслуговуванні [6].

При експлуатації уразливостей здійснюється порушення основних характеристик безпеки ресурсів КС: конфіденційності, цілісності і доступності. У цьому контексті за типом порушення зазначених характеристик уразливості бувають: К-типу (порушення конфіденційності ресурсів); Ц-типу (порушення цілісності ресурсів); Д-типу (порушення доступності ресурсів). Якщо в процесі експлуатації уразливості порушуються різні характеристики безпеки, то результуючий тип буде комбінований з основних, наприклад,

уразливість КЦД-типу, що порушує конфіденційність, цілісність і доступність ресурсів. Так, вищезгадувана уразливість у ПЗ обробки зображень у форматі WMF є уразливістю Д-типу.

За типом базового ресурсу, в якому міститься уразливість, уразливості поділяються на: ПВ-ресурсні (уразливості у протоколах взаємодії); ОС-ресурсні (уразливості в операційних системах чи системному ПЗ); ППЗ-ресурсні (уразливості у прикладному ПЗ) та ін. В свою чергу ОС-ресурсні уразливості поділяються на: Unix-ресурсні, Linux-ресурсні, Windows-ресурсні тощо. ППЗ-ресурсні уразливості бувають: ППЗ(Web-браузер)-ресурсні, ППЗ(e-mail)-ресурсні (уразливості у клієнтах електронної пошти), ППЗ(МП)-ресурсні (уразливості у програмі обміну миттєвими повідомленнями). Якщо стають уразливими інші типи ресурсів, то при класифікації уразливостей вводиться відповідне додаткове скорочення, а у випадку поширеності уразливості на декілька ресурсів тип визначається комбінацією базових ресурсів. Наприклад, у ПЗ та протоколах взаємодії мережі обміну миттєвими повідомленнями ICQ наявна уразливість, що дозволяє одному користувачу дізнатися, чи є в мережі інший користувач, який встановив режим невидимості. Така уразливість є ПВ-ППЗ(МП)-ресурсною.

За типом ініціюючого ресурсу, при взаємодії з яким стає уразливою програма, уразливості поділяються на: ФД-ресурсні (уразливості, що можуть бути використані при обробці певних файлів даних); К-ресурсні (уразливості, що ініціюються при обробці певного програмного коду); Д-ресурсні (уразливості, що можуть бути використані при обробці певних даних, що вводяться користувачем), та ін. В свою чергу ФД-ресурсні уразливості поділяються на: ФД(М)-ресурсні (ініціюються музичними файлами, наприклад, MP3-уразливість), ФД(В)-ресурсні (ініціюються відеофайлами, наприклад, AVI-уразливість), ФД(Г)-ресурсні (ініціюються графічними файлами, наприклад, JPEG-уразливість) чи навіть ФД(Т)-ресурсні (провокуються текстовими файлами, наприклад, DOC-уразливість) тощо. Якщо програми стають уразливими при взаємодії з іншими типами ресурсів, то при класифікації уразливостей вводиться відповідне додаткове скорочення, а у разі множинної дії тип визначається комбінацією ініціюючих ресурсів.

За наміром створення уразливості можуть бути навмисні (лазівки) та ненавмисні (дірки). Лазівки (*trap door*) є різновидом потаємного ходу (*back door*) у ПЗ, що створений розробником (у формі недокументованих властивостей) і використовується для обминання захисних механізмів в КС. Дірки (*loophole*) – недоробки, помилки у ПЗ, які дозволяють обминути процеси управління доступом [10].

Не важко помітити, що уразливості, які класифікуються за ознаковим принципом, можуть в кожному конкретному випадку при визначенні загального класу містити не тільки одну, але й більше компонент за будь-якою з ознак. Слід зазначити, що з появою нових уразливостей в початкових кодах, методів та засобів їх експлуатації ознаки запропонованої класифікації можуть бути розширені.

При практичному використанні класифікації, наприклад, така уразливість, як переповнення буфера при обробці спеціально сформованого графічного JPEG-файлу, може бути визначена як: код-екзогенна, автоматично спроможна (на цій уразливості паразитує вірус), дистанційована, елементарна, переповнення буфера, ЦД-типу, ППЗ(Web-браузер)/ФД(Г)-ресурсна (JPEG-уразливість), ненавмисна.

Запропонована класифікація може бути використана як для побудови комплексних систем аналізу захищеності ПЗ та систем виявлення визначених класів уразливостей в початкових кодах на базі системного підходу, що ґрунтується на низці окреслених ознак, так і для розробки вузькоорієнтованих систем, призначених для виявлення окремих підкласів уразливостей, наприклад, переповнення буфера. У цьому випадку описана розробка може стати базовою при формуванні вузьконаправленої класифікації для уразливостей переповнення буфера, що може бути використана для вибору й розробки відповідних методів та засобів побудови ЗПЗ. Запропонована класифікація може бути також узагальнена з метою розробки класифікації уразливостей у ПЗ.

Список літератури

1.*Хогланд Г., Мак-Гроу Г.* Взлом программного обеспечения: анализ и использование кода. – М.: Издательский дом "Вильямс", 2005. – 400 с.

2.*Weapons* For The Hunt: Methods for Software Risk Assessment. – OunceLabs Inc., 2004. – 14 p.

3.CERT/CC Overview Incident and Vulnerability Trends. – Pittsburgh: CERT Coordination Center, Software Engineering Institute, Carnegie Mellon University, 2003. – 221 p.

4.*Bishop M., Bailey D.* A Critical Analysis of Vulnerability Taxonomies (CSE-96-11), 1996. – 15 p.

5.*Robert C. Seacord, Allen Householder.* A Structured Approach to Classifying Security Vulnerabilities. Technical Note. – Pittsburgh: Software Engineering Institute, Carnegie Mellon University, 2005. – 39 p.

6.*Ховард М., Лебланк Д.* Защищенный код. – 2-е изд., испр. – Москва: Издательско-торговый дом "Русская Редакция", 2005. – 704 с.

7.*Эриксон Дж.* Хакинг: искусство эксплойта. – СПб.: Символ-Плюс, 2005. – 240 с.

8.*Sverre Huseby.* Innocent Code: A Security Wake-Up Call for Web Programmers. – John Wiley & Sons, 2004. – 246 p.

9.*Мак-Клар С., Шах С., Шах Ш.* Хакинг в Web: атаки и защита. – М.: Издательский дом "Вильямс", 2003. – 384 с.

10. *Бабак В. П., Корченко О. Г.* Інформаційна безпека та сучасні мережеві технології: Англо-українсько-російський словник термінів. – К.: НАУ, 2003. – 670 с.

11. *Пройдаков Э. М., Теплицкий Л. А.* Англо-русский толковый словарь по вычислительной технике, Интернету и программированию. – 3-е изд., испр. и доп. – М.: Издательско-торговый дом "Русская Редакция", 2002. – 640 с.

12. *Корченко А. Г.* Построение систем защиты информации на нечетких множествах. Теория и практические решения. – К.: "МК-Пресс", 2006. – 320 с.

Надійшла 17.05.2006

УДК 621.391:519.2

Алексейчук А.Н.

ВЕРХНИЕ ГРАНИЦЫ ПАРАМЕТРОВ, ХАРАКТЕРИЗУЮЩИХ СТОЙКОСТЬ НЕМАРКОВСКИХ БЛОЧНЫХ ШИФРОВ ОТНОСИТЕЛЬНО МЕТОДОВ РАЗНОСТНОГО И ЛИНЕЙНОГО КРИПТОАНАЛИЗА

Введение

Разностный и линейный криптоанализ [1–3] относится к числу наиболее известных общих методов криптографического анализа блочных шифров (БШ). За последние пять – семь лет произошли качественные изменения в основаниях этих методов, результатом которых стало более глубокое понимание их сущности, роли и положения в общей теории статистических методов криптоанализа БШ. В настоящее время разностный и линейный криптоанализ рассматривается, скорее, как различающий атаки, призванный выявлять определенные потенциальные слабости в конструкциях шифров, а не как метод практического взлома данных конструкций [4–6].

С точки зрения современного разностного и линейного криптоанализа, все блочные шифры естественным образом разделяются на классы марковских и немарковских шифров. Марковские (относительно той или иной групповой операции на множестве блоков шифруемых сообщений) шифры [2], к числу которых относятся DES, IDEA, Rijndael и многие другие БШ, образуют широкий класс блочных шифров, для которых удается построить общую содержательную теорию оценки и обоснования их теоретической или