Львівський державний університет безпеки життєдіяльності

Кафедра управління проектами, інформаційних технологій та телекомунікацій

Лекція

з курсу:

" Безпека інформаційно-комунікаційних систем"

на тему: **" Типові вразливості систем та причини їх появи."**

(для курсантів та студентів 5-го курсу спеціальності «Комп'ютерні науки»)

ПЛАН ЛЕКЦІЇ

1. Передумови виникнення вразливостей у комп'ютерних системах
2. Класифікація вад захисту
3. Помилки програмної реалізації системи: переповнення буфера та оброблення текстових рядків.
4. Люки.

ЛІТЕРАТУРА

1. **Герасименко В.А.** Защита информации в автоматизированных системах обработки данных. Т. 2. – М.: Энергоатомиздат, 1994. – 176 с.

2. **Зегжда Д.П., Ивашко А.М.** Как построить защищенную информационную систему. Том 1. - СПб: НПО «Мир и семья - 95», 1997. - 312 с.

3. **Мельников В.В.** Защита информации в компьютерных системах. – М.: Финансы и статистика: Эмектроинформ, 1997. – 368 с.

4. **Шатт С.** Мир компьютерных сетей: [пер. с англ.] – К.: BHV, 1996. – 288 с.

1. **Передумови виникнення вразливостей у комп'ютерних системах**

На цій лекції розглянемо причини появи вразливостей у комп'ютерних системах. Розгляд проводитиметься на основі практичного підходу до проблеми, тобто буде проведено аналіз конкретних випадків порушень безпеки і причин, що їх викликають. Такий підхід має велике значення для розроблення методів захисту, які дають можливість не лише протистояти наявним загрозам, але й виключати самі умови наявності вразливостей.

Докладніше розглянемо проблеми комп'ютерної технології оброблення інформації, що спричиняють уразливості. Вразливість — це нездатність системи протистояти певним впливам (випадковим або навмисним).

Перша і найголовніша проблема — бурхливий розвиток технології. Стрімке зростання обчислювальної потужності комп'ютерів і обсягів оброблюваних даних, а також розширення кола задач, які вирішують інформаційні системи, ускладнюють проведення повного і детального аналізу можливих вразливостей і виключення умов їх появи. Після того як користувачі набувають досвіду у використанні нової технології та розробляють адекватні заходи її захисту, ця технологія втрачає свою актуальність і привабливість. їй на зміну приходить нова технологія, використання якої також потребує певних знань і вмінь. Це стосується не лише технології програмування, а й технологій зберігання інформації і обміну даними між комп'ютерами.

Ситуація значно ускладнилася з появою персональних комп'ютерів. Але згодом відбулася ще одна технічна революція, пов'язана з виникненням глобальної комп'ютерної мережі — Інтернету. Застосування персональних комп'ютерів зробило мережу доступною для широкого кола користувачів. Через те концепції, які було покладено в основу функціонування Інтернету і побудови мережних протоколів для інформаційного обміну, виявилися неадекватними. Мережне середовище набуло зовсім не такого вигляду, яким його вбачали розробники. Сучасний стан Інтернету — це результат численних компромісів між ціною та якістю послуг, надійністю, безпекою і швидкістю обміну інформацією. Важливим чинником також є потреба в успадкуванні технологій і сумісностіз наявним обладнанням. Тому саме Інтернет став найбільш небезпечним джерелом загроз для інформаційних систем, які до нього підключені.

Основні причини появи вразливостей, що пов'язані з використанням комп'ютерних мереж у цілому та Інтернету зокрема.

По-перше, це спільне використання ресурсів і спрощення обміну інформацією між вузлами мережі. По-друге, суттєве ускладнення ПЗ, насамперед операційних систем. По-третє (це стосується багатьох сучасних мережних технологій, і не лише глобальних мереж), відсутність повної інформації про об'єкт і використання механізмів пошуку. Інтернет додає свої особливості: ненадійні джерела даних і величезна кількість зловмисників.

До фундаментальних причин наявності вразливостей належить також низька кваліфікація користувачів. На перший погляд, ситуація мала б бути протилежною: колись мало хто мав уявлення про комп'ютери, а сьогодні про них знають усі. Але йдеться не про обізнаність пересічного громадянина, а про кваліфікацію тих, хто реально працює на комп'ютерах. Нещодавно на комп'ютерах працювали лише спеціально підготовлені для того люди, а самі комп'ютерні системи було зосереджено в обчислювальних центрах, де підтримувався особливий режим доступу користувачів. Унаслідок широкого застосування персональних комп'ютерів їх почали використовувати школярі та пенсіонери, інженери й лікарі, люди творчих професій і оператори автоматизованих ліній на виробництві. Але переважна їх більшість не розуміється на питаннях захисту інформації та очікує від комп'ютерів легкого їх застосування й абсолютної надійності.

Разом із тим за легкістю використання комп'ютерів стоїть справжня складність технології. Наприклад, проста операція, на кшталт відкривання файлу з метою перегляду його вмісту, викликає послідовність дій, детальний опис якої може зайняти кілька сторінок тексту. Зокрема, за форматом файлу комп'ютер визначає, яку програму потрібно запустити для оброблення цього файлу, потім здійснюється запуск програми, для чого система надає необхідні ресурси, а вже тоді відкривається файл.

Суттєво впливає на безпеку ускладнення форматів подання даних, зокрема, сучасна тенденція об'єднання даних і програмного коду та вбудовування програмного коду (макросів, сценаріїв) у документи. Відкривання файлу даних (офісного документа, веб-сторінки) може викликати виконання певних дій, на які користувач не очікує.

Отже, некомпетентні дії з боку користувачів цілком імовірні. Це створює умови для існування специфічних уразливостей. Системи не здатні протистояти загрозам з боку зловмисників, якщо користувачі під їхнім впливом несвідомо виконують руйнівні дії. Розглянемо таку ситуацію: користувачу порадили відкрити консоль і ввести команду format c:, щоб прискорити доступ до жорсткого диска. Мабуть, знайдеться не дуже багато людей, які скористаються цією порадою. Проте більшість із них виконують такі (щоправда, дещо закамуфльовані) вказівки (особливо ті, що надходять електронною поштою), коли відкривають сумнівні файли, відвідують дуже сумнівні сайти в Інтернеті, повідомляють свої паролі (а інколи і реквізити кредитних карток) невідомим особам.

Часто виявляється, що спроектовані моделі безпеки не відповідають реальним системам. На стан захищеності інформації суттєво впливають такі типові чинники:

- модифікація архітектури системи;
- оновлення апаратних і програмних засобів та (або) їхніх можливостей;
- неналежна категорія та (або) кваліфікація персоналу;
- підключення до мережі (особливо глобальної).

Ще одна проблема, яка є прямим наслідком бурхливого розвитку технології, виникає через те, що нормативна та правова бази розвиваються не такими темпами, як змінюються методи оброблення інформації. Нові нормативні та правові акти, які регламентують певні сторони використання нової технології, з'являються поступово. Інколи трапляється так, що щойно прийняті акти потребують швидкого коригування, через зміни, що відбулися.

2. Класифікація вад захисту

Одна з необхідних умов створення захищених систем — проведення аналізу успішно здійснених порушень безпеки з метою їх узагальнення і класифікації задля виявлення причини і закономірності появи та існування вразливостей. Це дає змогу в подальшому, під час розроблення систем захисту, спрямовувати зусилля на усунення першопричин появи вразливостей, що допомагає ефективніше протидіяти загрозам. Розглянемо не вразливості загалом, а вади захисту — вразливості системи, що спричинені наявним у ній програмним кодом і дають змогу обійти впроваджені програмні засоби захисту.

2.1. Класифікація вад захисту за причиною їх появи

Вади захисту можуть бути внесені в систему навмисно чи випадково. Навмисність тут визначається не за мотивами дій конкретного користувача системи, який міг необережно запустити заражену вірусом програму чи «троянського коня», а за мотивами дій розробників програмних засобів. Тож до вад захисту, що вносять у систему навмисно, належать програмні закладки або спеціальне програмне забезпечення, здатне послабити засоби захисту. Вади захисту, що вносять у систему ненавмисно, — це помилки, яких припускаються розробники програмного забезпечення під час його проектування, реалізації, впровадження або супроводження.

2.2. Класифікація вад захисту за їх розміщенням у системі

Вади захисту можна класифікувати за ознакою їх розміщення в тих чи інших компонентах системи (рис. 1). Хоча вади захисту мають переважно програми, помилки, що спричиняють наявність вразливостей, можна зустріти і в апаратному забезпеченні. Докладно розглянемо лише програмні компоненти. Виокремимо програмні засоби таких трьох категорій:

- операційні системи;
- сервісні програми й утиліти;
- прикладне програмне забезпечення.

Розподілення на ці категорії є дещо умовним і залежить від конкретної термінології, обраної розробниками.

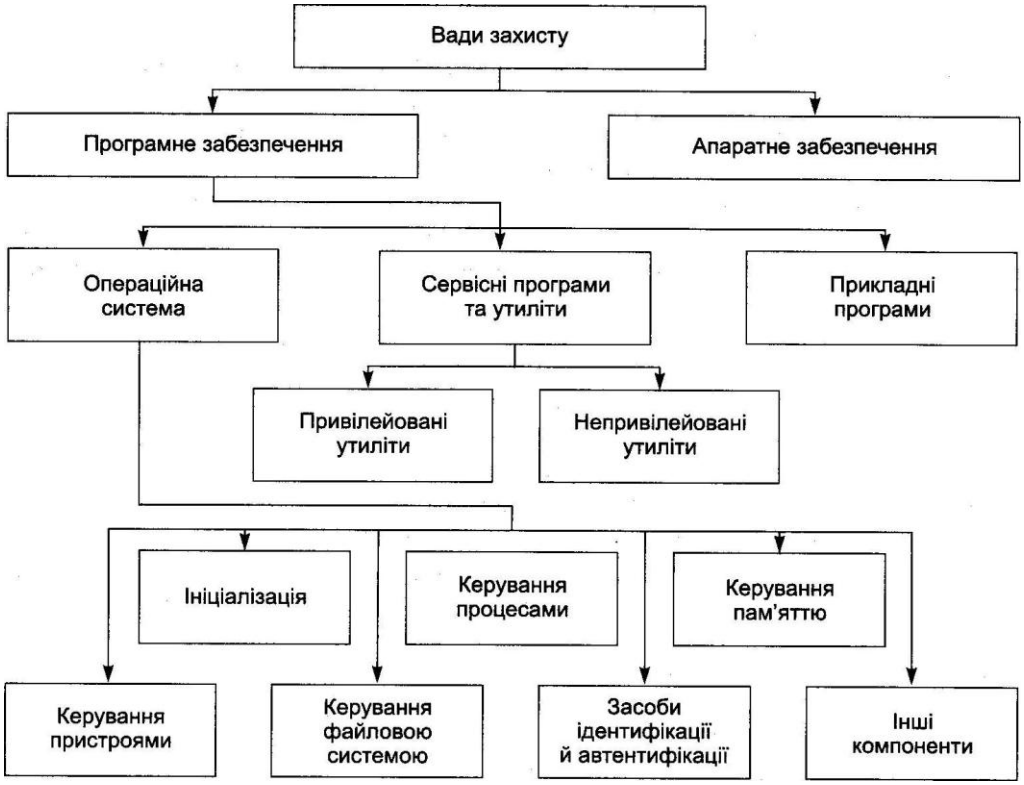

Рис. 1. Класифікація вад захисту за їх розміщенням у системі

Передусім потрібно чітко визначити, що саме розуміють під категорією сервісних програм. До цієї категорії належать компілятори, засоби налагодження програм, програмні засоби мережної взаємодії, системи керування базами даних, бібліотеки функцій. Головною ознакою таких засобів є те, що система, як правило, надає їм привілеї вищі, ніж у користувача, що з ними працює.

Наявні в операційній системі вади захисту автоматично спричиняють дуже суттєві вади захисту всієї інформаційної системи. Оскільки різні ОС можуть складатися з різних підсистем, ми розглянемо лише підсистеми, типові для більшості ОС і критичні з міркувань імовірності появи в них вад захисту, які призводять до порушення безпеки системи в цілому. До таких підсистем належать:

- засоби ініціалізації (завантаження) системи;
- керування процесами;
- керування пам'яттю;
- керування пристроями;
- керування файловою системою;
- засоби ідентифікації й автентифікації.

Помилки на етапі ініціалізації системи можуть спричиняти некоректну роботу пристроїв та (або) інших ресурсів системи. Наприклад, вони можуть стати недоступними через некоректне призначення повноважень доступу до цих ресурсів. Основу для розмежування доступу в системі створюють підсистеми керування процесами, пам'яттю та пристроями. Помилки в цих підсистемах надають порушнику можливість перевищувати повноваження, несанкціоновано захоплювати ресурси системи або отримувати повний доступ до інформаційних об'єктів. Те саме стосується і файлової системи. Окремо слід відзначити підсистему ідентифікації й автентифікації, на якій базується функціонування решти підсистем захисту. Помилки в цій системі можуть призвести не лише до некоректного виконання процедур ідентифікації й автентифікації, але й до розголошення ідентифікаторів і паролів.

Як було зазначено, сервісні програми й утиліти виконуються у системі з підвищеними привілеями. Тому помилки у таких програмах або навмисно впроваджені у них недокументовані функції спричиняють серйозні вади захисту системи в цілому. Є низка передумов появи помилок у сервісних програмах і утилітах. По-перше, вони, як правило,

досить складні, оскільки працюють за складними алгоритмами та взаємодіють із різними компонентами ОС, зокрема з драйверами пристроїв. По-друге, такі програми реалізують функції, які розширюють можливості ОС, а таке розширення інколи не відповідає розробленій для ОС моделі безпеки, зокрема не підтримує передбачені в системі обмеження. Таким чином, надійність захисту системи в цілому фактично залежить від коректності реалізації функцій захисту саме сервісних програм і утиліт.

Помилки, які спричиняють вади захисту в прикладному програмному забезпеченні, як правило, не здатні завдати помітної шкоди функціонуванню системного програмного забезпечення і призводять лише до збоїв процесу, що містить такі помилки. Але ті вади захисту, що є результатом навмисного впровадження шкідливих функцій у прикладні програми, вкрай небезпечні. Адже переважна більшість вірусів, «троянських коней», програмних закладок і спеціалізованих засобів атак функціонують саме як прикладні програми. Різниця полягає у тому, що спеціально впроваджені шкідливі функції використовують відомі вразливості у програмному забезпеченні, що дає їм змогу діяти з більшими повноваженнями, ніж повноваження прикладного процесу та користувача, який його ініціював.

2.3. Класифікація вад захисту за етапами їх появи

Наявність класифікації вад захисту за етапами їх появи має велике значення для здійснення аналізу передумов виникнення помилок та запобігання їм. Така класифікація тісно пов'язана з етапами життєвого циклу програмного забезпечення. Оскільки є різні моделі життєвого циклу та технології розроблення ПЗ, класифікації також можуть різнитися.

Рис. 2. Класифікація вад захисту за етапами їх появи

Життєвий цикл програмних систем складається з таких трьох основних етапів: розроблення, впровадження й експлуатація.

На першому етапі (якщо процес створення програмної системи розглядати спрощено) здійснюють:

- розроблення технічних вимог і специфікацій;
- розроблення алгоритмів;
- кодування.

У технічних вимогах визначають, що має робити програма. У специфікаціях описують, яким чином програма виконуватиме зазначені дії. Реалізація певних дій потребує розроблення алгоритмів. Відповідно до розроблених специфікацій і алгоритмів, створюють вихідні тексти програми та компілюють їх у програмний код. Іноді розроблення та компіляцію програм розглядають окремо, через наявність імовірності внесення специфічних помилок на кожному із цих етапів. Реальний процес кодування, як правило, проходить циклічно, тобто після написання чергового фрагмента вихідного тексту здійснюють його компіляцію і тестування, після чого робота триває.

Етап упровадження детально не розглядатимемо, лише зауважимо, що на цьому етапі відбуваються інсталяція системи, її налагодження у конкретному програмному й апаратному оточенні, а також випробування і (за потреби) атестація.

Етап експлуатації системи передбачає і супроводження, і застосування (що можна розглядати як окремі процеси і як один).

Під супроводженням системи мають на увазі її модифікацію та удосконалення шляхом виправлення наявних помилок, упровадження додаткових функцій та оновлення застарілих версій програм. Застосування системи — це експлуатація конкретної версії системи в певних умовах.

Розглянемо ці етапи докладніше, враховуючи імовірність появи на кожному з них специфічних вад захисту.

Розроблення технічних вимог і специфікацій

На цьому етапі навмисне внесення помилок малоймовірне, оскільки технічні вимоги і специфікації можна легко перевірити. Проте й тут виникають дві проблеми, які можуть спричинити появу вад захисту в кінцевому програмному продукті. Перша — це протиріччя між вимогами безпеки і загальними вимогами до функціональності системи. Виконати всі вимоги можна, лише приймаючи певні компромісні рішення, що сприяють послабленню безпеки. Типові приклади таких компромісів — спрощення процедур ідентифікації й автентифікації, розширення прав користувачів, збільшення квот на використання пам'яті, процесорного часу, дискового простору, кількості одночасних запитів користувачам і процесам тощо.

Друга проблема — невідповідність реальної системи технічним вимогам і розробленим на їх основі специфікаціям (а точніше, середовища, в якому система функціонуватиме). Така невідповідність виникає, коли стандартні проектні рішення тиражуються на різні системи і коли середовище системи зазнає змін на етапі його проектування. Найтиповішою і найнебезпечнішою з таких змін є підключення системи до глобальної мережі чи поява додаткових підключень в обхід запроектованих. Такі невідповідності можуть призвести до того, що всі обрані рішення із захисту втратять свою ефективність.

Розроблення алгоритмів

Алгоритми, що реалізують функції безпеки, також є потенційним джерелом вад захисту. Оскільки алгоритми підлягають перевірці, внесення навмисних помилок на цьому етапі малоймовірне, але ненавмисні помилки, наприклад, в алгоритмах роботи системи розмежування доступу цілком імовірні.

Кодування

Процедура створення вихідних текстів програм на основі розроблених специфікацій і алгоритмів надає широкий простір для виникнення вад захисту. На цьому етапі часто припускаються помилок. Більшість таких помилок можна виявити, проаналізувавши вихідні тексти програм разом зі специфікаціями. Проте дуже важко, майже неможливо, виявити ці помилки тестуванням скомпільованої програми, навіть якщо застосовувати для цього технологію «зворотної інженерії», тобто декомпіляцію.

Причинами появи помилок окрім складності програмного коду є особливості створення сучасних програмних систем, коли над проектом разом працюють багато виконавців, а також використання фрагментів уже готового коду (бібліотек). Останнє вимагає від програмістів не лише досконалого знання правил виклику тих чи інших функцій, але й особливостей їх реалізації, що не завжди відповідає реальності. Наприклад, найтиповіші помилки переповнення буфера, які надають зловмисникам можливість виконувати довільні команди на комп'ютері, як правило, виникають або внаслідок використання програмістами бібліотечних функцій з такою вразливістю, або через те, що ці функції викликаються іншими бібліотечними функціями, про що програмісти можуть не здогадуватися.

Вади захисту виникають на етапі розроблення тексту програм також через навмисне внесення недокументованих функцій — програмних закладок. Найчастіше програмісти за допомогою закладок створюють так звані люки, або чорні ходи (Backdoors), з метою спрощення процедур тестування і налагодження програми, а інколи і задля того, щоб у подальшому можна було скористатися ресурсами системи. Іноді вони вносять нешкідливі програмні закладки, які за виконання певних умов демонструють, наприклад, інформацію

про розробників. Але є й такі програмні закладки, які здійснюють шпигунську місію, приховано надсилаючи з системи конфіденційну інформацію, або виконують руйнівні дії. Зауважимо, що останнє фактично унеможливлюється у разі використання програмного забезпечення, розробленого «на замовлення» відомими розробниками, але цілком імовірне у програмах, отриманих із сумнівних джерел.

Компіляція

Окремо розглянемо помилки, що виникають у коді програм на етапі їх компіляції. За допомогою сучасних компіляторів можна робити численні настроювання, які мають відповідати розробленим специфікаціям і вихідним текстам програм. Таким чином можна обирати модель використання пам'яті, змінювати розмір сегментів (особливо стека), формат змінних за умовчанням і встановлювати додаткові перевірки параметрів під час виклику процедур, компіляції та у разі підключення додаткових модулів.

Компілятори також можуть містити недокументовані функції. Як приклад варто навести ідею Кена Томпсона — автора мови програмування С.

КенТомпсон запропонував увести в компілятор С програмну закладку, яка розпізнає вихідний код утиліти login, що виконує автентифікацію користувача в операційній системі UNIX і в процесі компіляції додає до цієї утиліти недоку-ментовану функцію, яка після введення спеціального таємного пароля надає користувачу привілейований доступ. Оскільки компілятор — це також програма, причому написана тією самою мовою програмування С, КенТомпсон запропонував упровадити в компілятор С ще одну програмну закладку, яка розпізнає вихідний текст компілятора і під час його компіляції додає до вихідного тексту обидві закладки.

Внесену компілятором програмну закладку майже неможливо виявити без трудомісткого аналізу машинних кодів. Оскільки вихідні тексти системних і прикладних програм UNIX були доступними, майже весь аналіз коду і пошук помилок виконувався не у від- компільованих кодах, а у вихідних текстах. Окрім того, всі нові версії компіляторів, реалізовані з використанням попередніх версій, також мали вносити ту саму програмну закладку. Таким чином, впроваджений люк із великою ймовірністю міг існувати протягом тривалого часу на всіх системах UNIX.

Основні висновки, що випливають із доповіді Кена Томпсона, зводяться до такого: не можна цілком довіряти програмі, написаній не власноруч, і не можна бути впевненим, що програмні продукти, навіть відомих і шанованих виробників* не містять програмних закладок.

Впровадження

На етапі впровадження системи виконується її налагодження в конкретному програмному та апаратному оточенні. На цьому етапі вади захисту можуть бути впроваджені через помилки в адмініструванні системи. їх спричиняє відсутність повної документації на систему, яка мала б надавати вичерпну інформацію про параметри, що можуть змінюватися під час інсталяції системи, та недостатній досвід персоналу в адмініструванні конкретної системи.

Супроводження

Під час супроводження системи також можуть бути внесені випадкові помилки, які спричиняють вади захисту. Такі помилки виникають переважно через недостатню обізнаність програмістів, що вносять зміни в систему, в деяких аспектах її функціонування. Внесення будь-яких змін потенційно загрожує безпеці системи. Єдиним ефективним способом захисту від цієї загрози є всебічне тестування системи після здійснення будь-яких її модифікацій (щоразу як нової системи).

Експлуатація

Цей етап має два джерела виникнення вад захисту. Перше — це помилки в адмініструванні системи. Добре спроектована система захисту зазвичай відстежує такі помилки адміністрування, як відключення окремих захисних функцій, звуження кола контрольованих об'єктів, надання підвищених привілеїв користувачам чи процесам. Адміністратори часто не зважають на попереджання системи і діють на свій розсуд, ігноруючи вимоги безпеки задля спрощення процедур адміністрування.

Другим джерелом появи вад захисту під час експлуатації системи є дія шкідливого програмного забезпечення, розробленого спеціально, щоб упроваджувати в систему програмні закладки. До таких програм належать, зокрема, комп'ютерні віруси, «троянські коні», мережні хробаки (докладніше про них йтиметься в розділі 6). Здебільшого запуск шкідливого програмного забезпечення здійснює персонал (авторизовані користувачі системи) через свою необачність або необізнаність. Іноді вади захисту цілеспрямовано впроваджують користувачі-порушники.

3. Помилки програмної реалізації системи

Вище було наведено різні класифікації вад захисту програмних систем. Тепер розглянемо типові помилки, які призводять до появи таких вад. У цьому підрозділі буде розглянуто лише помилки програмної реалізації, які виникають під час розроблення системи та її супроводження (рис. 3). Переважно ці помилки з'являються у кінцевому продукті через ненавмисні (випадкові) дії, хоча на певному етапі розроблення системи програмісти можуть додати (або, навпаки, вилучити)деякі функції навмисно задля спрощення процедур налагодження і тестування.

Розглянемо такі типові помилки (класифікацію помилок і більшість прикладів наведено за):

- помилки контролю припустимих значень параметрів;
- помилки визначення областей (доменів);

Рис. 3. Класифікація помилок, що виникають у процесі програмної реалізації системи

Помилки контролю припустимих значень параметрів

Такі помилки з'являються, коли певний механізм приймає хибне рішення щодо відповідності деякого параметра припустимим значенням. Це може також стосуватися кількості параметрів, їхнього типу, розміру тощо.

Помилки визначення областей

Типові помилки визначення областей виникають за наявності неконтрольованого доступу в захищені домени. Захищеними доменами можуть бути області пам'яті, файли на диску тощо. Наприклад, якщо після видалення об'єкта з пам'яті буде скасовано контроль доступу до області, яку він займав, але не видалено інформацію, що містилася в об'єкті, з'явиться помилка цього типу. Ще один приклад — надання можливості отримувати доступ на іншому рівні; це трапляється, коли доступ до файлів на диску контролюється системою розмежування доступу, а доступ до фізичних секторів диску — ні.

- *помилки послідовності дій;*
- *помилки ідентифікації й автентифікації;*
- *помилки перевірки границь об'єктів;*
- *інші помилки у логіці функціонування.*

Помилки послідовності дій

Ці помилки спричиняє асинхронний характер функціонування комп'ютерних систем. Не завжди можна організувати перевірку і виконання дій, що відповідають результату цієї перевірки, як неподільну операцію. Частіше перевірку виконує одна функція, а результат передається іншій. Помилка виникає, коли підміняють результат перевірки або ідентифікатор об'єкта (що ймовірніше), для якого було виконано перевірку. Наприклад, перевіряють права доступу до одного файлу, а здійснюють доступ до іншого.

Помилки ідентифікації й аутентифікації

Такі помилки трапляються доволі часто. Здебільшого вони виникають, коли підміняють автентифікаційну інформацію або виконують дії (створюючи для цього певні умови) без необхідної автентифікації суб'єкта та (або) об'єкта.

Помилки перевірки границь об'єктів

Помилки цього типу виникають через неконтрольованість виходу об'єкта за межі області пам'яті, виділеної для його зберігання. Це можуть бути текстові рядки, масиви, файли тощо. Саме до помилок цього типу належать найпоширеніші помилки переповнення буфера, які ми розглядатимемо далі.

Інші помилки у логіці функціонування

Є й інші помилки, які можуть бути використані зловмисниками. їх називають помилками у логіці функціонування системи і механізмів її захисту.

Помилки переповнення буфера

Помилки переповнення буфера (Buffer Overflows) — найпоширеніші з помилок, що призводять до появи вад захисту в програмних системах. Скориставшись цими вадами захисту, зловмисники можуть виконати будь-яку команду і отримати можливість контролювати всю систему.

Спочатку розглянемо основну ідею переповнення буфера. У будь-якому програмному коді програмісти організовують буфери для тимчасового зберігання й оброблення даних. Розмір буфера має бути таким, щоб дані, з одного боку, повністю у ньому вміщались, а з іншого, щоб буфер не був надто великим, оскільки тоді він марно займатиме пам'ять. Для копіювання даних у буфер переважно використовують бібліотечні функції. Якщо робота ведеться з рядками символів, то деякі функції не зважають на попереднє обмеження довжини і здійснюють копіювання до кінця рядка, тобто до символу, що є ознакою кінця рядка. До таких функцій належать, наприклад, *strcat(), strcpy(), sprintf (), vsprintf(), gets (), scanf ()*. Коли довжина рядка перевищує розмір буфера, копіювання триває, і частину рядка, що не вмістилась у буфері, буде записано замість даних, розташованих за його межами.

Зазначене повною мірою стосується мови програмування С, в якій ознакою кінця рядка є байт із нульовим значенням. У мові С така сама ситуація виникає під час роботи з масивами, оскільки автоматичну перевірку виходу за межі масиву не передбачено. Інші мови програмування можуть повністю або частково запобігати виникненню таких помилок.

Конкретні наслідки переповнення буфера залежать від того, яке значення мали втрачені або модифіковані дані та в якій області пам'яті було розміщено буфер: у статичній, динамічній пам'яті чи у стеку.

Помилки оброблення текстових рядків

У цьому підрозділі буде розглянуто типові помилки оброблення текстових рядків. Такі помилки траплялися дуже часто і були причиною вразливості багатьох комп'ютерних систем, підключених до глобальної мережі. Ми не будемо зосереджувати-ся на тому, які саме програмні продукти і під керуванням яких ОС були найбільш уразливими. Подібні помилки і нині трапляються майже на всіх системах, а також у глобальних і локальних мережах. Узагальнити характер цих помилок можна та-ким визначенням: некоректне оброблення непередбачених даних. Сприятлива для порушника ситуація, як і у випадку переповнення буфера, створюється тоді, коли вразлива програма виконується в системі з привілеями, вищими, ніж має користувач, від якого ця програма приймає дані.

Наявності цих помилок сприяє використання бібліотечних функцій, завдяки яким здійснюються введення й аналіз текстового рядка. Іноді програміст навіть не підозрює, що функція, яку він застосовує, у спеціальний спосіб обробляє певні символи.

Переспрямування введення-виведення

Переспрямування введення-виведення даних покладено в основу більш досконалих механізмів, на кшталт конвеєрів. Типова конструкція переспрямування введення-виведення має такий вигляд:

< file

Якщо в текстове поле ввести £ і Хе, то програмі буде передано ім'я файлу, а як-що < file — його вміст. Права доступу до такого файлу визначаються не права-ми користувача, який вводить цю конструкцію, а правами програми (або того користувача, від імені якого вона діє). Як і в ситуації з конвеєром, рекомендують проводити тестування поведінки програмних продуктів після введення таких конструкцій у будь-які поля введення.

4. Люки

Як уже зазначалося, люки (або «чорні ходи») — це недокументовані функції програмного забезпечення, які дають змогу здійснювати проникнення в систему. Програмісти можуть впроваджувати люки задля тестування та налагодження про-грам, а потім залишати їх у кінцевому продукті через неуважність або з метою по-дальшого використання ресурсів систем чи збирання інформації. Безумовно, ви-явлення люків у програмних продуктах завдає шкоди репутації розробника, тому їх не залишають або ретельно приховують. Крім того, люки можуть впроваджувати спеціально розроблені шкідливі програмні засоби (віруси, «троянські коні»)або зловмисники в результаті атаки.

Режим debug у програмі sendmail

Це один із найвідоміших люків, який, за твердженням розробників програми sendmail, було створено виключно задля її тестування і налагодження. Хоча такого люка не мало бути в розповсюджуваній версії програми, оскільки не відповідало вимогам безпеки, режим debug було залишено, і через нього вразливими стали безліч хостів в Інтернеті.

Основна задача програми sendmail - забезпечувати функціонування протоколу доставляння електронної пошти SMTP, тобто взаємодіяти з віддаленими хостами, зокрема здійснювати з'єднання через TCP-порт 25 для приймання вхідних листів. Докладніше про протокол і його вразливості йтиметься в далі; тут лише зазначимо, шо хоча програма sendmail функціонує в системі з великими привілеями, з нею може взаємодіяти будь-хто без будь-якої автентифікації, під-ключившись до TCP-порту 25.

Згідно з протоколом SMTP, можна визначати програми, що оброблятимуть поштові надходження. Таке визначення здійснюють у файлі псевдонімів (Aliases),який має право модифікувати лише привілейований користувач. Будь-який користувач може визначити у файлі .forward програму для оброблення власної пошти.

За допомогою непередбаченого протоколом SMTP режиму debug відправник пошти міг безпосередньо у полі адресата вказати програму, яка б цю пошту обробляла. Ллє відправником пошти міг бути будь-хто, програма для оброблення запускалася без перевірки і з тими правами, які мала програма sendmail. Фактично, це давало змогу порушнику виконувати на враженому комп'ютері будь-яку команду. Достатньо було одержувачу вказати | /bin/sh, і в тексті повідомлення він міг передати будь-яку команду або послідовність команд. Точніше, потрібно було ще видалити з пошти службові рядки (заголовки листа), для чого перед /bin/sh вказати придатний фільтр, наприклад редактор ed.