Безпека операційних систем і комп'ютерних мереж

Лекція 18

Безпека WWW

Основи веб-технології

- Веб-технологія в наш час є основною технологією в Інтернет
- Найголовніші її особливості:
  - використання гіпертекстових документів
  - застосування протоколу HTTP для обміну між клієнтом і сервером
- Програма, що виконується на боці клієнта, називається браузер
- У гіпертекстові документи можуть бути інтегрованими мультимедійні і програмні об’єкти
  - для демонстрації або виконання таких об’єктів використовуються вбудовані у браузер або підключені до нього модулі

Виконання програм на боці клієнта

- Основні типи програм, що можуть завантажуватись на комп'ютер користувача і виконуватись на ньому (на боці клієнта):
  - Java-аплети
    - технологія вже майже відійшла в минуле
  - програми, що написані на мові сценаріїв JavaScript (та деяких інших)
  - програмні компоненти ActiveX Controls
    - технологію завжди піддавали критиці, в наш час майже не застосовується
  - інтерактивні мультимедійні об'єкти Flash
    - технологію поступово витісняють, зокрема, вбудованими можливостями HTML 5
    - тим не менше, є її прихильники і аргументи на її користь

CGI інтерфейс

- Веб-сервер звертається до серверів, що реалізують прикладні функції, через інтерфейс CGI
  - Цей інтерфейс визначає
    - способи виклику Web-сервером прикладних програм або бібліотечних функцій
    - способи обміну інформацією з цими об’єктами
- CGI-програми, у свою чергу, можуть звертатись до інших серверів
  - наприклад, серверів баз даних

Обмін параметрами між клієнтом і сервером

- Клієнт і сервер за протоколом HTTP можуть обмінюватись параметрами
  - через запити і відповіді
    - клієнт може передавати серверу параметри методами **GET** і **POST**
  - через cookie
    - цей спосіб дозволяє серверу зберегти деякі дані про цього клієнта на боці клієнта у вигляді змінних в оперативній пам’яті або у спеціальних файлах

Динамічні сторінки

- Динамічні сторінки є реакцією Web-сервера на зміну зовнішніх умов
  - Зовнішні умови для Web-сервера – це дані, які сервер отримує від клієнтів за протоколом HTTP
    - параметри запитів **GET** і **POST**
    - деякі з заголовків, що відправляються клієнтом
    - cookies
  - Ці дані передаються програмам, що виконуються на боці сервера, і впливають на їх виконання
- Реакція сервера повинна бути повністю визначеною і документованою, тобто відомою наперед для будь-яких зовнішніх параметрів
  - Можуть існувати такі збіги зовнішніх умов (набори переданих параметрів) у сукупності із станом внутрішніх умов сервера (його тип і версія, ОС, СКБД, налаштовування самого сервера, ОС, файлової системи, змінні оточення, дані у файловій системі та у базі даних), які викликають непередбачену, як правило, несприятливу реакцію
  - У такому випадку можна говорити про вразливість системи до певних загроз і про можливість здійснення порушниками атаки

Уразливості клієнтського ПЗ

- Типові вразливості клієнтського ПЗ (браузерів) можна класифікувати як:
  - бінарні вразливості
    - наприклад, помилки переповнення буферу або можливість використання посилання на вже видалений об'єкт у пам'яті;
  - помилки, що дозволяють здійснити НСД до файлів на комп’ютері користувача;
  - помилки, що дозволяють підробляти чужі веб-сайти;
  - помилки контролю коректності коду сторінок.

Безпека Java

- Java широко застосовується у WWW як на боці клієнта, так і на боці сервера
  - Останнім часом застосування Java на боці клієнта значно скоротилось
  - Застосування Java на боці сервера, навпаки, поширюється
    - Це зумовлено насамперед вдалою оптимізацією багатопотокових застосунків, кращою ніж у конкуруючих технологій
- Великою перевагою технології Java є вбудована модель безпеки, яка пропонує рішення проблеми безпеки вже на рівні архітектури
  - Ця модель забезпечує гнучке керування доступом, яке легко налаштовувати, що дозволяє ефективно реалізовувати будь-яку політику безпеки

Уразливості Java

- У компіляторах і віртуальних машинах Java іноді виявляються помилки, що здатні негативно вплинути на безпеку комп'ютера (насамперед — бінарні вразливості)
  - Це вимагає від адміністраторів і розробників систем контролю повідомлень про виявлені помилки і застосування нових виправлених версій
- Завжди можна створити програму, яка не буде порушувати модель безпеки, але буде шкідливою з точки зору користувача. Наприклад, аплети Java можуть:
  - генерувати неприємні звуки з системного динаміка
  - не зупинятись при виході користувача з Web-сторінки, з якої аплет був завантажений
  - захоплювати значну частину системних ресурсів, наприклад, шляхом утворення великої кількості великих вікон на робочому столі тощо
- Аплети можуть взаємодіяти з іншими об'єктами, що завантажені у браузер
  - наприклад, із сценаріями Javascript, які мають більші можливості і менший контроль доступу до об'єктів комп'ютера

Чим небезпечні сценарії JavaScript

- Мова сценаріїв JavaScript була розроблена компанією Netscape і не має практично нічого спільного з Java крім співзвучної назви
- Сценарії можуть бути вписані безпосередньо у сторінку, або завантажуватись як зовнішні об’єкти
- Сценарії надають значні можливості для нападів на комп’ютери користувачів. Сценарії можуть:
  - звертатись до численних об’єктів ОС
  - відкривати нові вікна браузера
  - формувати веб-документи і демонструвати їх у вікнах
  - переадресовувати браузер з однієї сторінки на іншу
  - звертатись до змінних cookie
    - з міркувань безпеки вони не можуть читати і записувати cookie, які знаходяться не в тому домені, з якого був завантажений сценарій
    - втім, час від часу виявляються деякі помилки або послідовності дій, які дозволяють ці обмеження обходити
  - Сценарії можуть використовуватись для розповсюдження шкідливих програм
- У браузерах передбачена можливість заборонити виконання сценаріїв
  - Мало хто використовує цю можливість, оскільки переважна більшість сучасних сайтів в Інтернеті без виконання сценаріїв переглядати неможливо
- Сучасні антивірусні програми переглядають веб-сторінки у кеші браузера і шукають відомі їм шкідливі сценарії за сигнатурами

Підвищення ступеня захищеності клієнтського ПЗ (1/2)

- Свідомо обрати для себе той браузер, який задовольняє вимогам користувача щодо безпеки, функціональності, зручності та гнучкості налаштувань
  - Стежити за повідомленнями про виявлення вразливостей у браузері та усіх додаткових модулях, пов’язаних з ним
  - Своєчасно встановлювати виправлення та оновлені версії браузера.
- Налаштувати безпеку браузера
  - Встановлення чи відмова від встановлення певних модулів розширення
  - Захист сертифікатів та встановлення параметрів протоколів захищених з’єднань
  - Формування політики стосовно контенту і правил доступу до окремих вузлів

Підвищення ступеня захищеності клієнтського ПЗ (2/2)

- Не відвідувати сумнівні вузли
  - не слідувати усім посиланням, не перевіривши їх!
  - добре допомагають деякі розширення браузерів, що ведуть облік репутації сайтів
- Перевіряти весь завантажений контент антивірусним ПЗ з встановленими останніми оновленнями баз сигнатур
  - Антивірус має це робити автоматично
- Використовувати персональний брандмауер, який перевіряє не лише вхідні, а й вихідні з'єднання
  - вбудований брандмауер Windows XP останню вимогу не задовольняє
  - *Чи хтось дотепер використовує Windows XP???*

Основні загрози для сервера

- Веб-серверам, як і будь-якому іншому ПЗ, притаманні бінарні вразливості
  - Експлуатація бінарних вразливостей може спричинити
    - Виконання зловмисником довільного коду на сервері
    - Підвищення привілеїв (наприклад, отримання прав зареєстрованого користувача або root)
    - DoS атаку
- Більш характерними для веб-серверів є специфічні веб-уразливості
  - Ін’єкція вихідного коду
  - SQL-ін’єкція
  - Міжсайтове виконання сценаріїв (Cross-site scripting, XSS)
  - Експлуатація веб-уразливостей може призводити до:
    - Втрати конфіденційності (викрадення даних користувачів)
    - Несанкціонованої модифікації змісту (дефейс сайту)
    - DoS атаки

Типові веб-уразливості

- Ін’єкція вихідного коду
  - дуже поширена і одночасно одна з найнебезпечніших уразливостей сценаріїв, що виконуються на боці сервера
  - характерна для сценаріїв на PHP і на Perl
  - полягає у тому, що порушник отримує можливість впровадити і виконати довільний код на відповідній мові сценаріїв
- SQL-ін’єкція
  - уразливість, при якій порушник може впроваджувати свої дані, не передбачені розробниками веб-програми, у SQL-запит
- Міжсайтове виконання сценаріїв (Cross-site scripting, XSS)
  - уразливість, яку можуть мати сторінки, частину вмісту яких користувачі можуть змінювати, і ці зміни потім виводяться іншим користувачам, що відвідують сторінку
  - уразливість полягає у можливості впровадити сценарій, який потім буде виконуватись на комп’ютерах інших користувачів

Ін’єкція вихідного коду

- Ін’єкція вихідного коду (англ. – *Source Code Injection*) полягає у тому, що порушник отримує можливість впровадити і виконати довільний код на відповідній мові сценаріїв
  - Ця вразливість є дуже поширеною і одночасно є однією з найнебезпечніших
  - Вона характерна для сценаріїв на PHP і на Perl
- Ін’єкція вихідного коду PHP виникає внаслідок недостатньої перевірки змінних, що використовуються у таких функціях, як
  - **include()**
  - **require()**
  - та подібних

Ін’єкція вихідного коду – функція **include()** в PHP

- Функція **include()** в PHP підключає й виконує будь-який сценарій
- Якщо в якості аргументу функції передається URI, що вказує на файл на будь-якому HTTP чи FTP сервері, то такий віддалений файл буде завантажений за відповідним протоколом, після чого він буде виконаний
  - Для цього потрібні відповідні дозволи в налаштуваннях сервера)
  - Можливість включати зовнішні файли задається у налаштуваннях PHP
  - За умовчанням така можливість є
- Важливим обмеженням на підключення зовнішніх файлів може бути формат параметру, що передається функції **include()**
  - Якщо перед змінною, на яку порушник може впливати, стоїть будь-що, то тоді буде неможливо сформувати параметр функції, який був би коректним URI, що вказує на віддалений ресурс
    - наприклад, деякий префікс локального шляху: **include(“./profiles/ $user_id.php”)** не дозволяє вказати в URI протокол: **http://** чи **ftp://**
  - У такому разі немає вразливості *глобальної ін’єкції* вихідного коду

Ін’єкція вихідного коду - вразливість локальної ін’єкції

- Порушник може знайти можливість в якості сценарію підставити інтерпретатору PHP деякий файл, що знаходиться на сервері локально і є доступним для зчитування інтерпретатором.
- Слід враховувати, що розширення **.php** не є необхідним.
  - Інтерпретатор перегляне будь-який текстовий файл
  - Все, що знаходиться всередині програмних дужок PHP **<? ?>** або **<?php ?>**, буде сприйнято як команди PHP, і вони будуть виконані
  - Все, що знаходиться за межами таких дужок, інтерпретатор PHP вважає за простий текст, і просто виводить його без змін
  - Це надає потенційну можливість впровадити бажаний код у деякий локальний (на сервері) текстовий файл, доступний для зчитування
    - Це може бути навіть журнал реєстрації подій!

Ін’єкція вихідного коду в Perl

- Ін’єкція коду можлива і в Perl
- Для цього використовується функція **require()**
- На відміну від PHP, Perl дозволяє лише локальну ін’єкцію, і файл повинен бути повністю коректним сценарієм
- Доступ до цього файлу здійснюється з правами веб-сервера, і наявність права на зчитування є обов’язковою
- Однак, право на виконання файлу сценарію для його підключення функцією **require()** не потрібне
- Ні певне розширення імені файлу, ні навіть наявність стандартного першого рядка **#!/usr/bin/perl** також не є обов’язковими

Ін’єкція вихідного коду в Perl – помилка 500

- Для нормального відображенні у браузері сценарій Perl повинен обов’язково вивести HTTP-заголовок **content-type**, після якого йдуть два символи нового рядка:

  #!/usr/bin/perl

  print “Content-type: text/html\n\n”
- Причиною відсутності виводу заголовку **content-type** може бути те, що сценарій, до якого було здійснено звернення, насправді розроблявся як такий, що включається до іншого сценарію, і безпосереднє звернення до якого програмістом не передбачалось.
- Якщо такий рядок не буде виведений у HTTP-заголовок, або якщо перед цим рядком будуть видані інші рядки, які не є HTTP-заголовками, то результатом буде помилка 500 “Internal Server Error”
- Клієнт отримає повідомлення про помилку, а результати роботи сценарію клієнту надіслані не будуть
- При цьому у журналі реєстрації сервера також записується повідомлення про помилку
- Але насправді сценарій при цьому не переривається, а повністю виконується до самого кінця!

Ін’єкція вихідного коду в Perl – Створення процесу функцією open()

- Функція **open()** у Perl застосовується для відкривання файлів та зчитування їхнього вмісту
- При наявності в аргументі функції змінних, що доступні для зміни за протоколом HTTP, ця функція створює значну потенційну небезпеку
  - За відсутності належної перевірки HTTP-параметрів порушники отримують можливість щонайменше прочитати вміст не лише тих файлів, які передбачали розробники, а й деяких інших
  - В залежності від того, як саме передається параметр функції, доступними порушникам можуть стати будь-які файли, що дозволені для зчитування веб-серверу
- Функція **open()** у Perl підтримує конвеєр
- Якщо аргумент функції починається з символу “|”, то рядок, що йде за цим символом, сприймається як команда, яка виконується, і результат роботи якої виводиться замість вмісту файлу

Приклади роботи функції **open()** (1/2)

- Нехай сценарій **dumbtest.cgi** викликає функцію **open()** з аргументом **file**, що він отримує в якості GET параметра, після чого виводить вміст файлу
- Звернення клієнта за URI

  http://www.testserver.com/dumbtest.cgi?file=1.txt
  - видасть клієнту вміст файлу 1.txt (якщо такий файл існує у каталозі за умовчанням)
  - Якщо файлу не існує, помилки не буде – буде видана чиста сторінка
- Звернення клієнта за URI

  http://www.testserver.com/dumbtest.cgi?file=../../privat.txt
  - видасть клієнту вміст файлу, який знаходиться у каталозі, доступ до якого розробниками не передбачався (вважаємо, що такий файл існує)

Приклади роботи функції **open()** (2/2)

- Звернення клієнта за URI

  http://www.testserver.com/dumbtest.cgi?file=dumbtest.cgi
  - видасть клієнту текст самого сценарію
- Звернення за URI

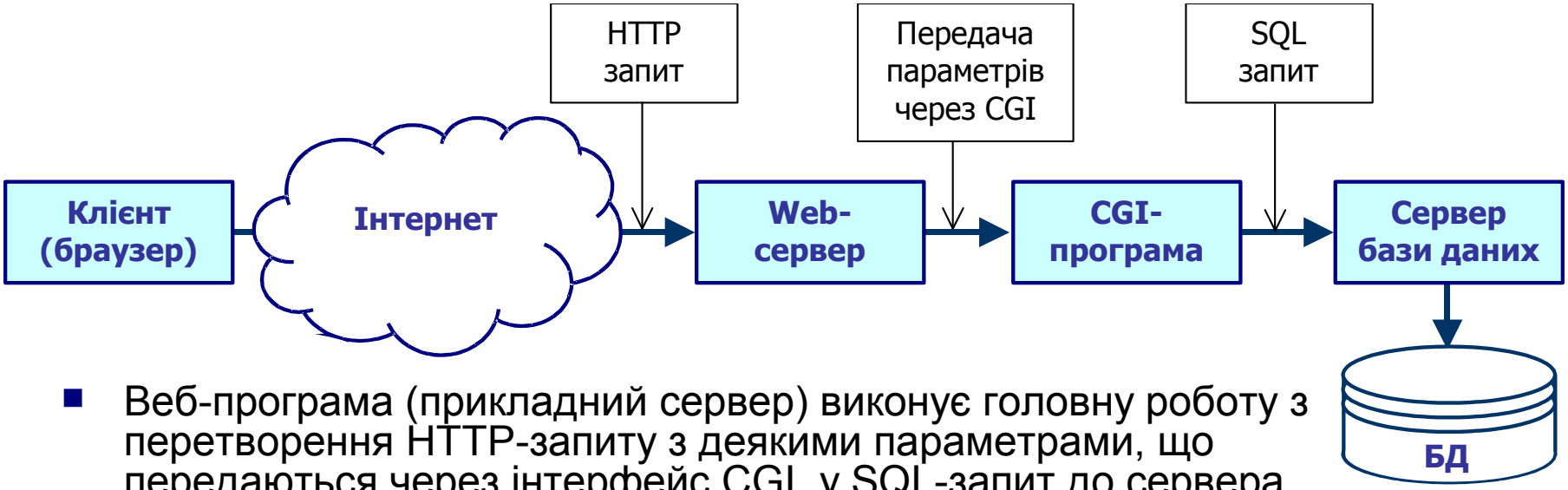
  http://www.testserver.com/dumbtest.cgi?file=|netstat+-an
  - видасть клієнту результат виконання команди **netstat**
  - тобто інформацію про мережні інтерфейси сервера, запущені сервіси і встановлені з'єднання
- Інші символи керування потоками введення-виведення також будуть працювати
- Наприклад, якщо існує **файл 1.txt**, то після запиту

  http://www.testserver.com/dumbtest.cgi?file=>1.txt
  - цей файл буде відкритий на записування (для цього необхідна наявність права Web-сервера на записування в цей файл), і вміст цього файлу буде знищено

Доступ до бази даних

- Дуже часто веб-програми забезпечують доступ користувачів з Інтернету до деякої бази даних
  - Саме у базі даних здійснюється зберігання інформації, з якої формуються динамічні веб-сторінки: стрічки новин, статті, інформаційні та рекламні матеріали, форуми, альбоми фотографій тощо
  - Переважна більшість матеріалів, які можна зараз побачити в Інтернеті, насправді зберігається не у вигляді файлів-сторінок, а у вигляді об’єктів у базах даних
- Доступ до баз даних здійснюється за допомогою SQL (англ. – Structured Query Language – структурована мова запитів)

Схема доступу до бази даних через WWW



- Веб-програма (прикладний сервер) виконує головну роботу з перетворення HTTP-запиту з деякими параметрами, що передаються через інтерфейс CGI, у SQL-запит до сервера баз даних
  - Саме на цю програму покладається левова частка роботи з перевірки коректності запиту і припустимості його параметрів
  - Саме вона формує SQL-запит
- Користувач взагалі не працює з SQL, у типовому випадку він лише активує деякі посилання на веб-сторінці (які часто оформлюються у вигляді кнопок, що їх треба “натискати”) і заповнює деякі поля у формах

SQL-ін’єкція (1/2)

- У типових реалізаціях веб-систем доступ до бази даних не обмежується лише зчитуванням
  - Найчастіше користувачі також мають чітко обмежені права додавати і редагувати деякі об’єкти у базі даних
    - Користувачі веб-ресурсів додають свої коментарі до статей, записи у книгах відвідувачів, створюють теми на форумах, додають у них свої повідомлення і мають можливість у подальшому їх редагувати і видаляти
  - Це означає, що налаштуваннями веб-сервера контроль доступу здійснити неможливо
- Сервер баз даних лише відпрацьовує ті SQL-запити, що до нього надійшли
- Саме прикладний сервер, який іноді називають “движком”, що реалізує усю логіку роботи відповідного веб-ресурсу, і повинен за параметрами HTTP-запиту формувати такий SQL-запит, який не суперечить політиці безпеки

SQL-ін’єкція (2/2)

- У багатьох випадках користувач може заповнити поля у формі таким чином, що в результаті формується непередбачений розробниками SQL-запит
  - Такий запит коректно відпрацьовується сервером бази даних і в результаті спричиняє порушення політики безпеки стосовно доступу до бази даних
    - Користувач (порушник) може впроваджувати дані, не передбачені розробниками веб-програми
  - У такому випадку кажуть про наявність вразливості SQL-ін’єкції (англ. – SQL source code injection)
- SQL-ін’єкція може бути критичною вразливістю в системі
- При цьому SQL-ін’єкція є також одною з найпоширеніших вразливостей

Як виникає SQL-ін’єкція

- Web-програма приймає через інтерфейс CGI параметри і використовує їх при формуванні SQL-запиту
- Наприклад, програма може отримувати параметр **id**, що надходить як GET-параметр HTTP-запиту, і формувати такий SQL-запит:

  SELECT * FROM table_1 WHERE id=‘$id’
  - Якщо в якості GET-параметра надійде значення id=123, то буде сформований нормальний SQL-запит, як це й передбачалось розробниками
- Порушник вручну змінив значення параметра у запиті таким чином:

  id=123’; SELECT user_id, pwd_hash FROM auth_data ‘
  - Результатом буде такий запит:

  SELECT * FROM table_1 WHERE id=‘123’; SELECT user_id, pwd_hash FROM auth_data ‘’
  - Ми ввели припущення, що фільтрація значення параметру id не здійснюється, і що порушник наперед знає про таблицю “auth_data”
- Порушник сформував власний SQL-запит, який не був передбачений розробниками, і який суперечить впровадженій політиці безпеки
  - Деякі сервери баз даних (MySQL, Oracle) не дозволяють поєднувати запити через символ “;”, але дозволяють зробити це іншими засобами
  - Інші сервери (PostgreSQL, MS SQL) дозволяють саме таку конструкцію

Чого може досягти порушник (1/3)

- Можливі практично будь-які дії з базою даних
  - SQL-запити дозволяють не лише отримувати деяку вибірку інформації, але й видаляти або модифікувати окремі поля, записи, або навіть цілі таблиці
- У програмах, які доступні зовнішнім користувачам за протоколом HTTP, найчастіше зустрічаються такі SQL-запити:
  - SELECT – вибір інформації з бази даних;
  - INSERT – додавання інформації до бази даних;
  - UPDATE – модифікація інформації у базі даних;
  - DELETE – видалення інформації з бази даних.

Чого може досягти порушник (2/3)

- Деякі сервери баз даних дозволяють роботу з файлами
  - Наприклад, в MySQL доступна функція **load_file()**, яка приймає в якості аргументу ім'я файлу і вертає його вміст
  - Також доступна конструкція SELECT … INTO OUTFILE <filename>, яка виводить результат запиту у файл, що заданий ім'ям з абсолютним шляхом <filename>
    - В обох випадках користувач повинен мати особливий привілей file_priv
    - Для того, щоби функція **load_file()** повернула вміст файлу, а не null, файл повинен бути доступним для зчитування усім користувачам
    - Файл, у який здійснюється вивід конструкцією SELECT … INTO OUTFILE на момент здійснення запиту не повинен існувати в системі
    - Після створення файлу, він стає доступним усім користувачам для зчитування і записування, але не для запуску на виконання
      - Як ми бачили раніше, це й не потрібно: якщо у файл виведено команди PHP, в подальшому він може бути підключеним і виконаним як сценарій при наявності лише права на зчитування
  - В PostgreSQL також існує можливість обміну інформацією між таблицями і файлами
    - Вона реалізується оператором COPY

Чого може досягти порушник (3/3)

- Шляхом SQL-ін’єкції можна здійснити DOS-атаку на сервер
  - Наприклад, в MySQL її можна здійснити, багаторазово викликавши функцію **benchmark(n, expr)**, яка n разів виконує заданий вираз **expr**
  - В MS SQL можливе виконання будь-якої системної команди шляхом виклику збереженої процедури **exec master..xp_cmdshell “cmd”**
    - Результат роботи команди при цьому порушнику не повертається, але сам факт виконання команди на сервері є надзвичайно небезпечним

Пошук вразливості SQL-ін’єкції

- Для виявлення і використання вразливості SQL-ін’єкції порушнику необхідно провести дослідження реакції програми (у тому числі повідомлень про помилки) на різні варіації усіх параметрів, що передаються їй методами GET, POST, та через cookie
- Для використання вразливості необхідно також визначити тип сервера бази даних
  - У цілому при наявності вразливості MySQL вимагає від порушника значно більших зусиль по дослідженню, ніж PostgreSQL, де експлуатація вразливості здійснюється порівняно легко
- Корисна книга:

  Низамутдинов М. Ф. Тактика защиты и нападения на Web-приложения. – СПб.: БХВ-Петербург, 2005. – 432 с.

Захист від SQL-ін’єкції

- Для унеможливлення цієї вразливості розробники Web-програм повинні включати ретельні перевірки типів усіх параметрів, що передаються
- Якщо очікуються дані числових типів, то необхідно
  - або впевнитись, що передані дані саме таких типів
  - або жорстко привести дані до очікуваного типу
- Якщо очікується рядок, необхідно ретельно контролювати, щоби користувач не міг вибратись у параметрі за межі рядка

Міжсайтове виконання сценаріїв (1/2)

- Міжсайтове виконання сценаріїв (англ. – *Cross-Site Scripting,* XSS) є дуже поширеною вразливістю
  - Також говорять і пишуть “міжсайтовий скриптінг”
- Цю вразливість можуть мати сторінки, частину вмісту яких користувачі можуть змінювати, якщо ці зміни потім виводяться іншим користувачам, що відвідують сторінку
  - Форуми
  - Чати
  - Системи обміну повідомленнями через веб-інтерфейс
  - Соціальні мережі
  - Сторінки, що містять статті або мультимедійні об’єкти, до яких користувачі можуть додавати свої коментарі

Міжсайтове виконання сценаріїв (2/2)

- Вразливість виникає тоді, коли не проводиться достатня фільтрація даних, що передаються користувачами
  - Порушник може впровадити такі дані, які можуть спричинити певну шкоду іншим користувачам
  - Для наявності вразливості необхідна можливість користувачу передавати не лише текстову інформацію, виведення якої шкоди нанести не може, але й посилання на інші документи й ресурси, а найголовніше – сценарії
- Через вразливу систему порушник отримує можливість виконати деякий сценарій на системі іншого користувача у контексті вразливої системи (сервера)

Активна і пасивна вразливості XSS

- Активна вразливість XSS — коли впроваджений зловмисником сценарій автоматично видається користувачам
  - Наприклад, код сценарію впроваджено в базу даних або у деякий файл, і він вставляється у сторінки, які сайт видає відвідувачам
  - Всі відвідувачі автоматично стають жертвами
- Пасивна вразливість XSS вимагає, щоби користувач сам виконав певні дії, наприклад пройшов по “отруйному” посиланню
  - Тобто, зловмиснику необхідно залучити соціальну інженерію, а жертві — піддатись на провокацію
  - Приклад “отруйного” посилання:

    http://www.site.com/page.php?var=<script>alert('працює!');</script>

Чого можна поробити

- Використовуючи вразливість типу XSS, порушник може досягти таких цілей:
  - здійснити “дефейс” сайту (зміну вигляду сторінки з метою дискредитації її власника, введення в оману користувачів, або простого хуліганства);
  - отримання параметрів (наприклад, cookie) від користувача;
  - збирання статистики щодо дій користувачів;
  - виконання неявних дій адміністратором.

Зміна вигляду сторінки (1/3)

- Найпростіші у виконанні зміни, які радикально змінюють вигляд сторінки, можна здійснити навіть без застосування JavaScript
  - Достатньо визначити стиль, що використовує непрозорий шар, який повністю перекриває оригінальну сторінку, і вивести цим стилем свою інформацію
    - З цією ж метою часто використовують <iframe>
  - Оригінальна сторінка буде надіслана користувачеві, але не буде йому видима
    - Переглянувши HTML-код сторінки, користувач може виявити оригінальну сторінку, а здійснивши редагування цього коду – і побачити
- Більш небезпечні зміни стосуються не вигляду, а наповнення сторінки, особливо – посилань, що на ній містяться

Зміна вигляду сторінки (2/3)

- Засобами JavaScript можна легко спрямувати користувача на зовсім іншу сторінку.
  - Наприклад, такий сценарій:

    <script Language=JavaScript>

    document.location.href="http://www.hacker.org/evilpage.html";

    <script>

    перенесе користувача на сторінку **http://www.hacker.org/evilpage.html**.
  - Користувач фактично опиниться на сайті зловмисника.
- У найгіршому випадку зловмисник відтворить дизайн тієї сторінки, з якої користувач був переадресований
  - Єдиною відмінністю (не дуже помітною для непідготованого користувача) буде інша адреса у полі адреси браузера
    - Якщо зловмисник готує серйозну атаку, він може зареєструвати доменне ім'я, подібне до атакованого сайту, і тоді помітити підміну буде ще важче.
- Це типова схема фішингу.
  - Такі дії можуть виконуватись для того, щоби отримати від користувачів атрибути доступу до сайту (імена і паролі), або іншу конфіденційну інформацію.

Зміна вигляду сторінки (3/3)

- Ще один варіант підміни сторінки – організація різних DOS-атак
  - Дуже легко організувати DOS-атаку на клієнта:
    - прості сценарії JavaScript можуть відкривати величезну кількість дуже великих і невидимих (прозорих) вікон, повністю з'їдаючи ресурси процесора і пам'яті
  - Також сценарії можуть ініціювати звернення до іншого сервера, який у такому випадку стане ціллю розподіленої DOS-атаки:
    - ненавмисно атакувати його почнуть усі користувачі, що зайшли на змінену зловмисником сторінку.

Отримання параметрів від користувача

- Припустимо, порушник бажає отримати деякі параметри комп'ютера користувача
  - JavaScript дозволяє миттєво отримати усі параметри cookie у контексті того сайту, з якого сценарій був завантажений
    - Для цього достатньо звернутись до властивості cookie об'єкту document
  - Також JavaScript дозволяє відправити отримані дані на будь-яку адресу
- Головною проблемою для порушника є необхідність здійснити це непомітно
  - Найпростіший варіант – відправити дані методом POST на деякий веб-ресурс, що контролюється порушником
    - Але при цьому користувач буде переадресований з того сайту, з якого він отримав модифіковану порушником сторінку, безпосередньо на сайт порушника, що майже напевно буде помічено
  - Більш досконалий варіант – засобами JavaScript відкрити нове вікно, в якому звернутись до веб-ресурсу порушника, передавши йому усі необхідні дані
    - Нове вікно можна зробити мінімального розміру, і розташувати за межами екрану, а після відправлення порушнику даних – закрити його.

Збирання статистики

- Порушник має можливість збирати статистику про відвідувачів сторінки, яка має вразливість типу XSS
  - Для цього йому достатньо вставити посилання на деякий об’єкт (наприклад, картинку, або фоновий звук), що знаходиться на сайті порушника
  - Клієнтські програми кожного з відвідувачів, отримуючи сторінку, будуть звертатись по цей об’єкт до сайту порушника, де цілком прозоро може збиратись статистика
  - Довести факт несанкціонованого збирання статистики досить важко
    - для цього необхідно мати доступ до внутрішніх налаштовувань сервера порушника
  - Для збирання статистики порушник може так налаштувати свій Web-сервер, що запити до файлів з розширенням, наприклад, .gif (зображення), будуть оброблятись як сценарії PHP. А сам сценарій вже буде надсилати користувачу необхідну картинку
    - На такому принципі побудовані так звані Інтернет-жучки
- В результаті порушник легко отримує таку статистику:
  - IP-адреси відвідувачів (навіть якщо сама система їх приховує);
  - час відвідань (може допомогти у встановлені відповідності між умовними іменами користувачів та їхніми IP-адресами);
  - тип і версія браузера, тип і версія ОС;
  - з якої саме сторінки користувач звернувся до сайту порушника
- Зібрана статистика може дозволити визначити конфіденційні дані окремих користувачів

Виконання неявних дій адміністратором

- Це досить складна атака, яка, тим не менше, може бути здійснена
- Ідея полягає у тому, щоби надіслати адміністратору веб-сторінку, модифіковану таким чином, що адміністратор непомітно для себе виконає деякі дії, потрібні порушнику
  - Наприклад, якщо вразливим ресурсом є форум, порушник у такий спосіб може руками адміністратора здійснити деструктивні дії, недоступні звичайним користувачам (видалення повідомлень або цілих тем)

Захист від уразливостей типу XSS

- Вище ми припускали, що порушник може безперешкодно впровадити свій JavaScript у сторінку сайту
  - Насправді так бути не повинно!
- Фільтрація повинна проводитись
  - за ключовими словами (script і т.п.)
  - за окремими символами, які становлять найбільшу небезпеку (одинарні та подвійні лапки, кутові дужки).
- Часто взагалі забороняють додавати будь-які теги HTML
  - Замість них часто використовуються так звані BB-коди, які використовують не кутові, а квадратні дужки
    - З точки зору HTML вони не мають жодного спеціального значення
    - Веб-програма (наприклад, движок форуму або чату) заміняє ті з них, які розпізнає і вважає припустимими, на відповідні HTML-теги, а ті, що не розпізнає – залишає без змін (тоді вони виводяться як простий текст) або взагалі відкидає
- Але, як показує практика, порушники дуже вигадливі у формуванні “неочікуваних” рядків, які в результаті некоректно обробляються веб-програмами
  - Інформацію про конкретні можливості проникнення можна знайти на багатьох сайтах в Інтернет

Приклад вразливості типу XSS: спарені теги (1/3)

- Якщо один BB-тег вставити в другий, вони обидва будуть перетворені у теги HTML
  - Оскільки один з них знаходиться всередині іншого, структура HTML порушується
  - Таке порушення структури може дозволити ін'єкцію сценарію
  - Вразливість спарених тегів існувала, зокрема, у популярному форумному движку phpBB
- Тег

  [COLOR=color]colored_text[/COLOR]

  перетворюється в HTML таким чином:

  colored_text
  - В якості color можна використовувати будь-які символи, крім ";"
    - це захищає від ін'єкції сценаріїв JavaScript у **style**
  - Символ апострофу перетворюється в його HTML-код '
    - це виключає можливість закрити одиночні лапки і вийти з атрибуту **style** у тіло тегу.

Приклад вразливості типу XSS: спарені теги (2/3)

- Якщо ж у якості color указати інший BB-тег, наприклад **[IMG]**, то конструкція виявлялась коректною з точки зору движка форуму:

  [COLOR=[IMG]http://addr.com/file.jpg[/IMG]]colored_text[/COLOR]

- Зазначена конструкція перетворювалась у такий HTML-код:

  **<span style='color:'> colored_text **

- Движок не фіксував очевидної некоректності цієї конструкції через те, що спочатку обробляв тег **[COLOR=color]**, і не вважав неприпустимим рядок color, що містив в собі BB-тег, а вже потім обробляв тег **[IMG]**
- Як можна помітити, лапки з HTTP-тегу **img** закривають лапки атрибуту **style**: **style='color:<img src=‘**

Приклад вразливості типу XSS: спарені теги (3/3)

- При цьому фрагмент **http://addr.com/file.jpg** опинився ззовні атрибутів, але всередині тегу, що зумовлює можливість ін'єкції сценарію JavaScript
- Незначна модифікація конструкції

  [COLOR=[IMG]http://addr.com/=`file.jpg[/IMG]]`
  style=background:url(javascript:alert()) [/COLOR]
- дає в результаті HTML-код

  <span style='color:<img src='http://addr.com/=`file.jpg' border='0'
  alt='user posted image' />'>`
  **style=background:url(javascript:alert()) **
- Під час обробки цього коду браузер, намагаючись звернутись за URL, що заданий для фону, виконає впроваджений сценарій JavaScript