Міністерство освіти і науки України

Головне управління освіти і науки Полтавської облдержадміністрації

Полтавське територіальне відділення МАН України

Кременчуцьке районне наукове товариство учнів «Мала академія наук»

Секція: інформатика та
обчислювальна техніка

Підсекція: безпека інформаційних
та телекомунікаційних систем

РЕАЛІЗАЦІЯ ЗАБЕЗПЕЧЕННЯ ЗАХИСТУ ІНФОРМАЦІЇ У СИСТЕМІ КЕРУВАННЯ ЗМІСТОМ ВЕБ-САЙТУ

Роботу виконав:
Кукса В'ячеслав Вікторович,
учень 11 класу
Кременчуцької ЗОШ І-ІІІ ступенів
№ 17 «Вибір» імені М.Г. Неленя

Науковий керівник:
Лагута Світлана Миколаївна,
вчитель інформатики
Кременчуцької ЗОШ І-ІІІ ступенів
№ 17 «Вибір» імені М.Г. Неленя,
спеціаліст вищої категорії

Науковий консультант:
Лисенко Тетяна Іванівна,
викладач інформатики
Кременчуцького педагогічного
училища ім. А.С. Макаренка,
Заслужений вчитель України

Полтава - 2011

Полтавське територіальне відділення МАН України

Кременчуцьке районне наукове товариство учнів «Мала академія наук»

ТЕЗИ

до науково-дослідницької роботи

«Реалізація забезпечення захисту інформації у системі керування змістом веб-сайту»

учня 11 класу ЗОШ №17 “Вибір” імені М.Г. Неленя

Кукси В’ячеслава Вікторовича

Науковий керівник: вчитель інформатики, Лагута Світлана Миколаївна

В останні роки, з розвитком комерційної і підприємницької діяльності збільшилося число спроб несанкціонованого доступу до конфіденційної інформації, а проблеми захисту інформації виявилися в центрі уваги багатьох вчених і спеціалістів із різноманітних країн.

Важливою умовою інформаційної безпеки стає безпека в комп’ютерних мережах. Тому захист інформації - важливе і першочергове завдання при проектуванні веб-сайтів.

Оскільки в минулому році автор працював над проблемою використання системи керування контентом як засобу для побудови веб-сайту, цілком природною стала проблема захисту інформації у розробленій системі. Представлена робота є логічним продовженням дослідження, яке було представлене у минулому році під назвою «Розробка системи керування змістом веб-сайту».

Об’єкт дослідження – використання системи захисту інформації і розмежування доступу як засобу надійної безпеки веб-сайту.

Предмет дослідження – напрями забезпечення конфіденційності інформації у CMS.

Мета дослідження – визначити різні підходи до забезпечення захисту інформації при побудові веб-сайтів та розробити власну програму з урахуванням вироблених критеріїв.

Завдання дослідження:

1. Проаналізувати основні загрози безпеці веб-сайтів.
2. Визначити основні структурні складові систем захисту інформації та їх функції.
3. Проаналізувати наявні системи безпеки у комп'ютерних мережах та визначити напрями забезпечення захисту інформації.
4. На основі визначених напрямів розробити власну програму для забезпечення захисту інформації на веб-сайті під керуванням CMS.

Теоретична частина базується на нормативно-правовій базі для організації заходів захисту інформації в системах телекомунікацій, яка відображена у Законі України "Про інформацію".

У ході виконання теоретичної частини дослідження були визначені основні структурні складові систем захисту інформації та їх функції, проаналізовані основні види загроз веб-сайтам: SQL-ін'єкція, ін'єкція коду, XSS і вироблена стратегія захисту інформації на веб-сторінках. Ядро системи керування вмістом веб-сайту MyCMS написано мовою програмування PHP 5. Сценарії, написані цією мовою, вразливі до двох основних видів атак – SQL-ін'єкції та ін'єкції коду, тому практична розробка орієнтована саме на ці види атак. Ядро системи безпеки представляє собою бібліотеку API мовою PHP та інтегрований з редактором MyCMS засіб для керування розмежуванням доступу через візуальний інтерфейс.

Особливості розробки: захист від атак типу SQL-Injection забезпечується тим, що параметри, отримані від користувача не фігурують безпосередньо у запитах до бази даних, тому невірні значення параметрів не здатні порушити логіку запиту. Такий підхід зменшує швидкодію системи, але виключає, повністю або частково, необхідність перевірок вхідних параметрів, які при найбільшій ретельності розробника часто вдається обійти.

Практичне значення роботи – застосування на веб-сайтах, створених за допомогою системи MyCMS, розробленій у ході виконання автором дослідницької роботи у минулому році.

ЗМІСТ

ВСТУП

В даний час, в Україні, в зв'язку з входженням у світовий інформаційний простір, швидкими темпами впроваджуються новітні досягнення комп'ютерних і телекомунікаційних технологій. Системи телекомунікацій активно впроваджуються у фінансові, промислові, торгові і соціальні сфери. У зв'язку з цим, різко зріс інтерес широкого кола користувачів до проблем захисту інформації. Тривалий час методи захисту інформації розроблялися тільки державними органами, а їхнє впровадження розглядалося як виключне право тієї або іншої держави. Проте, в останні роки, з розвитком комерційної і підприємницької діяльності збільшилося число спроб несанкціонованого доступу до конфіденційної інформації, а проблеми захисту інформації виявилися в центрі уваги багатьох вчених і спеціалістів із різноманітних країн.

Оскільки основний інформаційний обмін сьогодення оснований на інформаційній технології, то важливою умовою безпеки стає безпека в комп’ютерних мережах. Тому захист інформації - важливе і першочергове завдання при проектуванні веб-сайтів.

Оскільки в минулому році автор працював над проблемою використання системи керування контентом як засобу для побудови веб-сайту, цілком природною стала проблема захисту інформації у розробленій системі.

Об’єкт дослідження – використання системи захисту інформації і розмежування доступу як засобу надійної безпеки веб-сайту.

Предмет дослідження – напрями забезпечення конфіденційності інформації у CMS.

Мета дослідження – визначити різні підходи до забезпечення захисту інформації при побудові веб-сайтів та розробити власну програму з урахуванням вироблених критеріїв.

Завдання дослідження:

1. Проаналізувати основні загрози безпеці веб-сайтів.
2. Визначити основні структурні складові систем захисту інформації та їх функції.
3. Проаналізувати наявні системи безпеки у комп’ютерних мережах та визначити напрями забезпечення захисту інформації.
4. На основі визначених напрямів розробити власну програму для забезпечення захисту інформації на веб-сайті під керуванням CMS.

Проаналізувавши наявні загрози системі безпеки у комп’ютерних мережах та визначивши основні напрями захисту інформації, розроблено власний програмний засіб для забезпечення конфіденційності інформації на веб-сайті.

Практичне значення роботи – застосування на веб-сайтах, створених за допомогою системи MyCMS, розробленій у ході виконання автором дослідницької роботи «Розробка системи керування змістом веб-сайту» у минулому році.

РОЗДІЛ 1
Засоби з захисту інформації в інформаційних системах

1.1 Актуальність проблеми безпеки інформації

Інформаційні технології охоплюють методи збору, обробки, перетворення, зберігання і розподілу інформації. Протягом тривалого часу ці технології розвивалися на мовній і "паперовій" літерно-цифровій основі. У наш час інформаційно-ділова активність людства зміщується до кібернетичного простору. Цей простір стає реальністю світової спільноти і визначає перехід до "електронного" розвитку інформаційних технологій. Електронний обмін інформацією дешевше, швидше і надійніше "паперового". Наприклад, ціна листа з Каліфорнії в Нью-Йорк складає поштою - USD 0,29 (час доставки 2-3 дні), тоді як плата за факс - USD 1,86, за телекс - USD 4,56 і, нарешті, за E-mail - всього USD 0,16, причому час пересилки - години - хвилини.[1]

Інформаційні процеси висувають на перший план, поряд із задачами ефективного опрацювання і передачі інформації, найважливішу задачу забезпечення захисту інформації. Це пояснюється особливою значимістю для розвитку держави його інформаційних ресурсів, зростанням вартості інформації в умовах ринку, її високою уразливістю і значним збитком у результаті несанкціонованого використання. [2]

У багатьох країнах порушення безпеки в системах опрацювання і передачі інформації приносять великі втрати. Вони найбільше значні в системах телекомунікацій, що обслуговують банківські і торгові заснування. У США, наприклад, збитки від несанкціонованого проникнення в ці системи оцінюються в десятки мільйонів доларів.

Про ступінь небезпеки електронних злочинів можна судити по тим витратам на засоби безпеки, що вважаються припустимими і доцільними. По оцінках спеціалістів США, загальні витрати на захист банківського або іншого фінансового заснування можуть скласти усього 510 тис. доларів. Проте надійна система захисту,

що обслуговує до 80 тисяч користувачів, коштує не менше 15 млн. доларів, причому в цю суму входить тільки вартість апаратних і програмних засобів.[1]

Наслідки від несанкціонованого одержання інформації мають самий різноманітний масштаб: від необразливої прокази до фінансових витрат великих розмірів.

Останнім часом різко почастішали випадки розкрадання програм у комп'ютерних мережах. Ці розкрадання прийняли характер епідемії: на кожну законну копію програми, що має скільки-небудь широке поширення, існує декілька копій, отриманих незаконним шляхом.

Оскільки сучасний інформаційний обмін оснований на інформаційній технології, то важливою умовою безпеки стає безпека в комп'ютерних мережах. Порушення цієї безпеки називають комп'ютерним злочином. Збиток від комп'ютерних злочинів дуже великий (так крадіжки в INTERNET за 1997 рік привели до збитку USD 8млрд). [1]

Перерахуємо основні загрози для комп'ютерних мереж:

1. Читання інформації (порушення конфіденційності).
2. Порушення цілісності (інтегральності) інформації.
3. Блокування доступу до об'єктів і ресурсів комп'ютерних мереж.

Загрози діляться на пасивні і активні. До пасивних загроз відноситься підслуховування (зчитування) інформації та аналіз трафіка (ідентифікація користувачів і визначення інтенсивності інформаційного обміну). До активних загроз відноситься модифікація даних, створення фальшивих даних під чужим і'ям, введення вірусів і програмних закладок, блокування доступу до ресурсів комп'ютерних мереж.

1.2 Нормативно-правова база для організації заходів захисту інформації в системах телекомунікації

В обчислювальних машинах є велике число лазівок для несанкціонованого доступу до інформації. Ніякий окремо взятий спосіб захисту не може забезпечити адекватну безпеку. Надійний захист може бути гарантований лише при створенні

механізму комплексного забезпечення безпеки як засобів опрацювання інформації, так і каналів зв'язку [3].

Нормативно-правове забезпечення організації і проведення заходів щодо захисту інформації являє собою сукупність законів, нормативних актів і правил, що регламентують як загальну організацію робіт, так і створення і функціонування конкретних систем захисту інформації. В даний час в Україні, як і в інших країнах СНД, нормативно-правова база захисту інформації знаходиться в стадії формування. [2]

Закон України "Про інформацію" установлює загальні правові основи одержання, використання, поширення і збереження інформації. Відповідно до цього закону: інформація – це документовані або привселюдно оголошені відомості про події і явища, що відбуваються в товаристві, державі або навколишньому природному середовищі.

Закон закріплює право особистості на інформацію у всіх сферах суспільного і державного життя України, а також систему інформації, її джерела, визначає статус учасників інформаційних відношень, регулює доступ до інформації і забезпечує її охорону, захищає від помилкової інформації.

Метою Закону України "Про захист інформації в автоматизованих системах" є встановлення основ регулювання правових відношень по захисту інформації в автоматизованих системах за умови дотримання права власності громадян України і юридичних осіб на інформацію і права доступу до неї, права власника інформації на її захист, а також установленого чинним законодавством обмеження на доступ до інформації [3,4].

Захист інформації – сукупність організаційно-технічних заходів і правових норм для попередження заподіяння збитку інтересам власника інформації і осіб, що користуються інформацією.

Аналіз показує, що перелік загроз безпеки інформації дуже різноманітний [1]:

- перехоплення даних - огляд даних несанкціонованим користувачем;

- аналіз трафіка - огляд інформації, що стосується зв'язку між користувачами (наприклад, наявність/відсутність, частота, напрямок, послідовність, тип, обсяг обміну і т.д.);
- зміна потоку повідомлень, видалення повідомлень або порушення загального порядку повідомлень у потоку;
- відмова користувача від повідомлення, заперечення відправником свого авторства в пред'явленому йому одержувачем повідомленні або заперечення одержувачем факту одержання їм повідомлення;
- маскарад - прагнення порушника видати себе за деякого іншого користувача з метою одержання доступу до додаткової інформації або нав'язування іншому користувачу помилкової інформації;
- порушення зв'язку, недопущення зв'язку або затримка термінових повідомлень.

1.3 Програмні засоби захисту інформації

Програмні засоби - це спеціальні програми і програмні комплекси, призначені для захисту інформації в інформаційних системах, наприклад PGP – комп’ютерна програма, а також бібліотека функцій, що дозволяє виконувати операції шифрування та цифрового підпису інформації; модулі сервера Apache: mod_security та mod_rewrite; CrackLib – бібліотека для перевірки даних у PHP-сценаріях.

З засобів програмного забезпечення систем захисту необхідно особливо виділити програмні засоби, що реалізують механізми шифрування (криптографії), та генерації електронно-цифрового підпису, наприклад, PGP.

У загальному випадку в систему забезпечення захисту інформації можуть бути включені [1]:

1) служба таємності даних – може бути використана для захисту переданих даних від можливості проведення аналізу інтенсивності потоків даних між користувачами;

2) служба аутентифікації – призначена для підтвердження того, що в даний момент зв'язку користувач є дійсно тим користувачем, за якого він себе видає;

3) служба цілісності даних – забезпечує доказ цілісності даних у процесі їхньої передачі, тобто забезпечує захист переданих повідомлень від випадкових і навмисних впливів, спрямованих на зміну переданих повідомлень, затримку і знищення повідомлень;

4) служба керування доступом – забезпечує захист від несанкціонованого доступу до інформації, що утримується в віддалених банках даних, або від несанкціонованого використання мережі;

5) служба цілосності інформації – забезпечує доказ цілісності повідомлення, прийнятого від відповідного джерела і знаходиться на збереженні;

6) служба доставки – забезпечує захист від спроб зловмисника порушити зв'язок.

Найбільш поширені способи захисту, метою яких є дозволити обробляти форму з даними виключно «живій» людині, а не боту (бот – спеціальна програма, що виконує автоматично і за заданим розкладом певні дії, видаючи себе за користувача):

- способи захисту, що потребують дій від користувача;
- способи, що не потребують дій від користувача;
- реєстрація користувачів та модерація коментарів.

1. Способи захисту, що потребують дій від користувача. Система захисту від спам-ботів CAPTCHA - «спотворені» символи (цифри, букви) на форумах, гостьових книгах, при реєстрації нових користувачів, відправці SMS через Internet, які потрібно розпізнати і ввести з клавіатури. CAPTCHA - це комп’ютерний тест, який використовують для того, щоб визначити, ким є користувач Web-сайту: людиною чи комп’ютером [5].

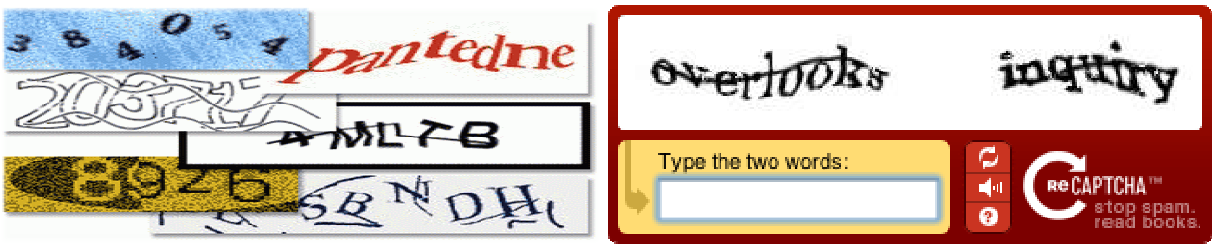

Рис. 1.1 Приклади CAPTCHA [6]

2. Способи, що не потребують дій від користувача. Ці способи захисту є більш дружніми до відвідувача сайту, ніж різного роду CAPTCHA і тому при їх застосуванні зручність в користуванні сайтом для відвідувача зростає. Система може як обмежувати кількість запитів, так і намагатись відрізнити людину від бота за непрямими ознаками в поведінці. Головна відмінність бота-програми від людини - це підхід до Web-сторінки як до послідовності тексту і тегів, тоді як пересічна людина бачить кінцевий результат (візуальний образ) і код сторінки її мало цікавить. Більшість способів, які не потребують активних дій від користувача при відправленні Web-форми, базуються на тому, що середньостатистичний бот, на відміну від повноцінного браузера, не навчений інтерпретувати CSS, JavaScript, Flash, тощо. Розглянемо деякі способи, які не потребують втручання користувача:

- обмеження по частоті повідомлень. Суть методу: необхідно стежити, щоб з однієї і тієї ж IP-адреси не було більше ніж, наприклад, десять повідомлень в хвилину. Він має сенс тоді, коли примушувати користувача вирішувати CAPTCHA не можна, а захистит є необхідним. Обмеження кількості запитів з однієї IP-адреси - це доволі простий спосіб, малоефективний при достатній підготовленості зловмисника;
- блокування за часом завантаження форми. Захистом може бути блокування при обробці повідомлення на певний час, що пройшов між завантаженням форми і її відправкою - людині, на відміну від бота, потрібен якийсь час на введення даних (який, як правило, більший за 1-2 секунди);
- блокування повідомлень за ключовими словами. Фільтр працює так: до публікації не допускаються повідомлення, що мають в тексті непристойні слова, образливі, лайливі фрази, або шкідливий програмний код;
- зміна імен полів, які беруть участь в передаванні даних. Більшість спам-роботів шукають на сторінці поля із стандартними іменами, наприклад, «name», «email», «mail» і тому подібне. Щоб доставити боту деяку незручність, краще називати поля нестандартно. Або можна, наприклад, поле «електронна пошта» назвати «name», а «ім’я» - «email», з розрахунку на те, що бот прийме рішення про суть полів по атрибуту name тегу <input>;

- створення полів-приманок. Спам-бот шукає поля «name», «email» і тому подібні. Створюємо приховане поле, не hidden, а, наприклад, приховане засобами CSS. Звичайний відвідувач не бачить це поле і природно не заповнює його. Спам-робот заповнить це поле. Форма повинна оброблятися тільки, якщо приховане нами поле буде порожнім. Серед полів-приманок звичайно мають бути поля, які б бачив користувач, але їм слід дати нестандартні імена;
- блокування повідомлень за розміром екрану. Даний спосіб захисту працює за простою схемою: якщо при програмному визначенні у відвідувача нема розмірів (ширини, висоти) екрана, то форму намагається надіслати бот, отже її не слід обробляти;
- побудова форми за допомогою JavaScript. Цей метод робить аналіз форми ще складнішим: боту необхідно знайти і виконати код JavaScript, щоб отримати поле, яке слід заповнити. Можна, наприклад, заховати поле в JavaScript-код так [5]:

```
<form action="..." name="addMsg" method="post">
<script language="JavaScript" type="text/javascript">
...
document.write('<inp'+'ut n'+'ame="a'+'ntis" t'+'ype="hi'+'dden"
va'+'lue="1'+'10'+'4">');
...
</script>
</form>
```

А тоді на стороні сервера слід перевіряти чи існує $_POST[’antis’] та, чи вона дорівнює 1104.

3. Реєстрація користувачів та модерація коментарів. Суть цього методу полягає в тому, що, наприклад, коментарі в Гостьовій можуть залишати не всі відвідувачі, а лише ті, що зареєстровані на сайті. Реєстрація є бар’єром для масової розсилки спаму. Модерація коментарів полягає в тому, що публікуються тільки ті думки відвідувачів, що прямо схвалені адміністратором Web-ресурсу. Мінуси

такого підходу: адміністраторові сайту потрібно буде приділяти увагу і витрачати час на перевірку користувачів, на видалення/редагування некоректних повідомлень, на затвердження коментарів, тощо. Крім того, існують незручності для відвідувачів - не кожен захоче реєструватись, щоб залишити свою думку про опубліковану статтю або чекати схвалення свого запису в Гостьовій [6].

В цьому розділі розглянуті деякі найбільш популярні способи захисту сайту від атак спам-роботів. Описані в ній приклади не гарантують на 100% того, що сайт ніхто не «зламає». Завжди, навіть в популярних і довершених системах є вузькі місця, приклад тому випадок, коли на сайтах по всьому світу була знайдена груба помилка, яка дозволяла виконувати довільні запити до бази даних мовою SQL (Structured Query Language - структурована мова запитів), і яка отримала назву SQL Injection.

1.4 Аналіз загроз веб-сайтам

Щоб зламати веб-портал зловмисник збирає інформацію, необхідну і достатню для його зламу:

- перевірка використання на веб-сайтом сценаріїв, написаних власниками сайту або розроблених комерційними організаціями;
- детальний розгляд сценаріїв, які потребують введення даних користувачем;
- розгляд того, як сценарії обробляють введені дані;
- розгляд того, як фільтруються введені дані, щоб обійти ці фільтри;
- використання універсального web-фільтру, який продивляється HTTP-заголовки повідомлень.

SQL-ін’єкція – вбудовування вільних SQL-команд, у результаті якого змінюється логіка запиту до бази даних. Це становить загрозу, бо таким чином можна поцупити з бази даних конфіденціальну інформацію. Приклад – через помилки web-інтерфейсу у різних провайдерів не раз крали бази з логінами та паролями користувачів. Успішність атаки SQL-ін’єкція не залежить від

використання для написання web мови програмування – чи то PHP, Perl, або ASP. Якщо сценарій працює з базами даних, а перевірка вхідних параметрів відсутня, то завжди є можливість приєднання SQL-коду.

Приклад вразливого сценарію:

```
<?php
…
$data=mysqli_query($link,”SELECT * FROM profiles WHERE id=”.$_GET[“id”]);
…
?> [1]
```

У данному прикладі значення поля id буде порівнюватися зі значенням параметра id переданому через URL.

Якщо в якості значення параметра вказати, наприклад, 10, то чесний користувач побачить свій профіль. Але, якщо вказати в якості параметра id рядок 10 OR UserName= “Administrator”, то запит до бази даних набуде вигляду SELECT * FROM profiles WHERE id=10 OR UserName= “Administrator”. Після виконання такого запиту відобразиться не лише запис з id=10, але й запис у якому поле UserName=“Administrator”. Тобто після такого запиту хакер побачить всю інформацію про обліковий запис “Administrator”.

Php-ін'єкція - один зі методів злому веб-сайтів, що працюють на PHP, який полягає у виконанні стороннього коду на серверній стороні. Потенційно небезпечними є стандартні функції PHP [7] :

eval();

fopen();

exec();

require_once();

include_once();

include();

require();

create_function();

Php-ін'єкція стає можливою, якщо вхідні параметри приймаються і використовуються без перевірки.

Загроза XSS. В багатьох XSS вже давно немає ніякого скриптингу. Всяка можливість впливу на віддаленого користувача, яка реалізовується через незахищений сайт віддаленим хакером, і буде XSS.

XSS-атаки відрізняються моделлю передачі даних між клієнтом, сервером та хакером. В цілому на сьогоднішній день відомо три основні моделі [1]:

1) XSS DOM. У цій моделі вразливість сайту полягає у тому, що не серверний сценарій, а саме клієнтський JavaScript вставляє в код HTML сторінки дані, отримані в URL, через об’єкти Document Object Model (DOM). Тому користувачу достатньо лише перейти за посиланням, яке містить XSS-вектор хакера, і вміст веб-сторінки буде змінено і в неї безпосередньо на машині клієнта буде вставлений код з тіла вектора.

Приклад вразливої сторінки:

```
<HTML>
…
<TITLE>Welcome!</TITLE>
…
<SCRIPT>
  var pos=document.URL.indexOf("name=")+5;
  document.write(document.URL.substring
  (pos,document.URL.length));
</SCRIPT>
...
</HTML>
```

Приклад вектора:

```
http://www.vulnerable.site/welcome.html?
name=<script>alert(document.cookie)
</script>
```

2) Класичний XSS. Найбільш поширена модель атак. Використовувана вразливість полягає в тому, що при недостатній фільтрації вхідних параметрів

серверним сценарієм, тіло XSS-вектора потрапляє до результуючого HTML, CSS, або JavaScript-коду безпосередньо у браузер клієнта.

3) Збережений XSS. Фактично, ця модель атак пов’язана з перманентним розміщенням параметрів для скрипта, які передаються від хакера серверу, безпосередньо в базі даних сервера й подальшій їх видачі відвідувачам сайту (найчастіше це форуми, блоги і т.п.). Ця модель, з точки зору зловмисника, має дві великі переваги: вона не потребує розсилки даних хакером користувачу (цю роботу виконує сервер і користувач нічого не запідозрить), і надає можливість створювати, так званих, XSS-хробаків – на комп’ютері кожного відвідувача вразливого сайту виконається хакерський код, і цей код може сам себе розміщувати на інших сторінках сайту.

Приклад XSS-хробака мовою JavaScript:

```
function HTTPRequest (url)
{
// гілка для XMLHttpRequest object
if (window.XMLHttpRequest) {
req = new XMLHttpRequest();
req.onreadystatechange =processReqChange;
req.open("GET", url, true);
req.send(null);
// гілка для IE/Windows ActiveX version
} else if (window.ActiveXObject) {
req = new ActiveXObject("Microsoft.XMLHTTP");
if (req) {
req.onreadystatechange =processReqChange;
req.open("GET", url, true);
req.send();
}
HTTPRequest("http://vulnerable-blog.com/vulerable-
script.php?vulnerable-arg=<iframesrc=http://hacker-site/xss.js>");
```

Типова XSS-атака складається з наступних стандартних етапів: []

- пошук вразливості;
- вибір методу передачі інформації або впливу на користувача через XSS-вектор;
- створення додаткових інструментів для підтримки XSS Proxy, віддалено розміщених файлів та інше;
- пошук способу розповсюдження XSS-вектора;
- створення ефективного XSS-вектора;
- вмілого експлуатування отриманих даних.

Класифікація основних загроз і їх характеристик представлена у таблиці 1.1

Таблиця 1.1

Класифікація загроз веб-сайтам				
Назва	**Загроза**	**Розповсюдження**	**Складність захисту**	**Об'єкт атаки**
Code injection	найвища	низька	низька	скрипт (з привілеями веб-сервера)
SQL Injection	висока	середня	середня	база даних
XSS	середня	висока	висока	кінцевий користувач

Відсотковий розподіл загроз веб-сайтам наведений на рис. 1.2

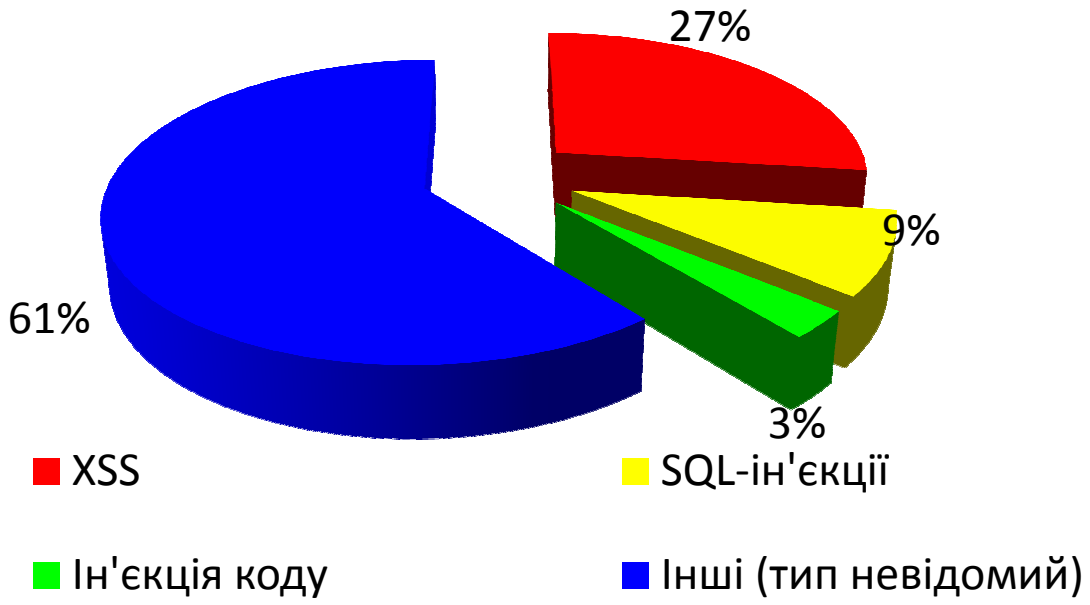

Рис. 1.2 Відсотковий розподіл загроз веб-сайтам

Проаналізувавши наявні загрози системі безпеки у комп’ютерних мережах та визначивши напрямки захисту інформації, пропоную власну програмну розробку для забезпечення конфіденційності інформації на веб-сайті.

РОЗДІЛ 2
Практична розробка програмного засобу

2.1 Стратегія захисту

Використовуємо чотири рівня захисту інформації:

1. Використання спадаючого списку замість рядка введення даних.
2. Аудит введення даних у форми на боці клієнта за допомогою JavaScript.
3. Аудит введених даних на боці сервера.
4. Запит на сервер про правильність введення даних.

Розглянемо плюси та мінуси даної стратегії. Вона, безумовно, відсторонить зломщиків – дилетантів, які не вміють користуватися нічим, окрім браузера. Відзначимо ще один приємний ефект – покращення юзабіліті сайта, бо при невірному заповненні форми користувач ще до відправки форми отримає повідомлення. Мінус у захисті – його можна обійти дуже просто: відключення в браузері JavaScript усуває один з рівнів захисту, а ще краще – працювати не через браузер, а за допомогою сценарію.

Проаналізувавши потенційну загрозу скрипта, визначаємо, що форма, яку заповнює користувач найбільш вразлива. При використанні методу POST дані поступають сценарію на стандартний вхід, а при використанні методу GET інформація передається в адресний рядок:

URL

http://www.example.ru/index.php?variable=value

Для сценарію, написаного на PHP, всі вищезгадані методи отримання зовнішньої інформації рівнозначні, тому можна використати масив $_REQUEST. Окрім того, скрипт отримує дані ще й від бази даних. Зловмисник міг отримати доступ до бази даних і занести туди довільну інформацію, наприклад з використанням SQL-ін’єкції, або через міжсайтовий скриптинг – XSS. Тому фільтри перевірки потрібно накладати на дані і при отриманні від користувача і при передачі цих даних самому користувачу [8].

Забезпечення захисту інформації на боці сервера можна проілюструвати схемою на рис. 2.1

Для обмеження довжини рядка даних, який вводить користувач та перевірки всіх полів в PHP використовуємо функцію is_numeric (для перевірки на число). Для перевірки на коректність адреси електронної пошти можна використовувати регулярні вирази:

PHP:

```
eregi("^[A-Z0-9._%-]+@[A-Z0-9._%-]+\.[A-Z]{2,6}$", $email)
```

Для перевірки пароля можна використати бібліотеку CrackLib. Це дозволить порівняти пароль із словником і виконати ще кілька перевірок.

PHP:

```
// Завантажуємо словник
$dictionary = crack_opendict('/usr/
local/lib/pw_dict');
// Перевіряємо пароль
$check = crack_check($dictionary,
.Q6g$b87gHjn5_4t5sdf!23HLayi');
// Отримуємо результат перевірки
$res = crack_getlastmessage();
echo $res; // 'strong password'
// Закриваємо словник
crack_closedict($dictionary);
```

Ще один підхід – збереження IP користувача – зробить неможливим роботу користувача з іншого комп'ютера до завершення його сеансу на сайті [8].

Фільтрація спец символів – важливий елемент захисту від атак типу SQL-injection та Code-injection.

PHP:

```
function escape_smart($value)
{
```

```
if (get_magic_quotes_gpc()) {
$value = stripslashes($value);
}
$value = mysql_real_escape_string($value);
 return $value;
}
```

Найбільш використовувані функції PHP для екранування спеціальних сиволів наведені у таблиці 2.1[1]

Таблиця 2.1

Приклади екранування символів

Назва	**Опис**
string addslashes (string str)	додає до символів апостроф, лапки, зворотньому слеш і нульовому байту слеш
string quotemeta (string str)	додає слеш до символів \+*?[^]($)
string htmlspecialchars (string string)	перетворює в HTML-сутність амперсанд, лапки, апостроф, знаки «Більше» та «менше»
string htmlentities (string string)	аналог htmlspecialchars, але використовує HTML-сутності
string mysql_real_escape_string (string unescaped_string)	екранує рядок для використання в mysql_query

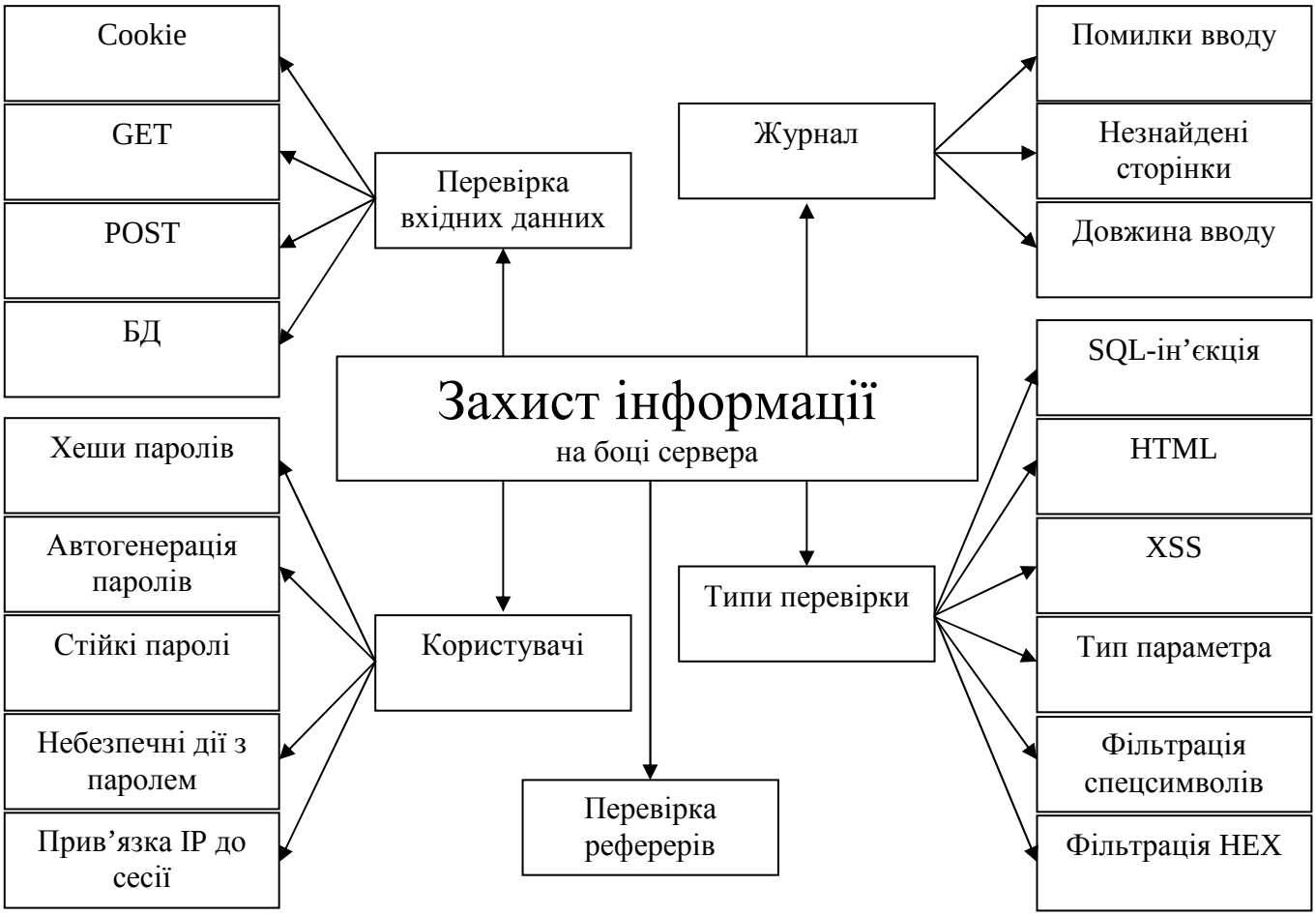

Рис. 2.1 Схема захисту інформації на веб-сайті

2.2 Реалізація захисту інформації на веб-сторінках

Ядро системи керування вмістом веб-сайту MyCMS написано мовою програмування PHP 5. Сценарії написані цією мовою вразливі до двох основних видів атак – SQL-ін'єкції та ін'єкції коду. У ході теоретичного дослідження була вироблена стратегія захисту веб-сайтів від різних видів атак, яка знайшла застосування у власній розробці – системі MyCMS.

Захист від атак типу SQL-Injection забезпечується тим, що параметри, отримані від користувача не фігурують безпосередньо у запитах до бази даних, тому невірні значення параметрів не здатні порушити логіку запиту.

Такий підхід зменшує швидкодію системи, але виключає, повністю або частково, необхідність перевірок вхідних параметрів, які при найбільшій ретельності розробника часто вдається обійти.

Це можна продемонструвати на прикладі коду функції LogIn, яка відповідає за вхід зареєстрованого користувача на сайт під керуванням MyCMS [8].

```
function LogIn($login,$password,$mysqllink)
 {
   $is_logined=-1;
    /* Обчислюємо MD5-хеш рядка-пароля */
   $password_md5=md5($password);
   /* Отримуємо інформацію про всіх зареєстрованих користувачів з бази даних MySQL за допомогою запиту без параметрів */
   $result=mysqli_query($mysqllink,'SELECT * FROM users');
    /* У циклі перевіряємо наявність у базі даних облікового запису користувача з даними логіком і паролем */
   while ($resultarray=mysqli_fetch_assoc($result))
    {
       if (($resultarray["user_login"]=$login)and
         ($resultarray["user_password_md5"]=$password_md5))
       {
        /* Якщо користувач знайдений у базі даних, розпочинаємо новий сеанс, та вносимо дані користувача до глобальної змінної сесії */
         session_start() ;
         $_SESSION["user_id"]=$resultarray["user_id"];
         $_SESSION["user_pass_md5"]=$password_md5;
         $is_logined=0;
         $current_user_id=$resultarray["user_id"];
         $current_user_md5=$password_md5;
         $user_logined=true;
         break;
```

```
        }
    }
    /* Звільняємо пам’ять, виділену для набору даних */
    mysqli_free_result($result);
      return $is_logined;
}
```

Дана функція приймає три параметри $login – рядок-логін, введений користувачем, $password – рядок-пароль користувача, $mysqlink – дескриптор з’єднання з базою даних MySQL, яке було відкрите сценарієм, що викликає функцію. Оскільки вхідні параметри не фігурують у запиті до бази даних, їх перевірка виявляється непотрібною.

Атака типу «ін’єкція коду» може використовувати передачу системі керування змістом веб-сайту шляхів до файлів-шаблонів веб-сторінок. Тому в системі MyCMS введені псевдоніми шаблонів. База даних системи містить таблицю псевдонімів шаблонів, у якій кожен псевдонім заданий записом з двома полями: ідентифікатором шаблону та повним шляхом до файлу шаблону.

Сценарій - шаблонізатор отримує від користувача в якості параметра ціле число – ідентифікатор шаблону. Шлях до файлу береться із бази даних, до якій розробником сайту попередньо записуються псевдоніми. Нижче наведений скорочений код сценарія-шаблонізатора з докладними коментарями.

```
<?php
    …
    /* Превіряємо, чи був переданий ідентифікатор шаблону */
    if (isset($_GET["template"]))
      {
        /* Перевіряємо, чи є отриманий параметр числом */
        if (!is_numeric($_GET["template"]))
        {
          die('<b><h1>Warning</h1></b> <hr> <h3>Parameter "template" must be an integer number!</h3><hr>MyCMS 2.0.1');
        }
```

```
            /* Отримуємо шлях до файлу шаблону з бази даних */
            $template_id=$_GET["template"];
            $dataset=mysqli_query($link,"SELECT template_file FROM templates
WHERE template_id=".$template_id) or die("<h1><b>Query error:
".mysqli_errno($link)."</b></h1> <hr> <h3>Cannot show page</h3><hr>MyCMS
2.0.1");;
            $datarow=mysqli_fetch_assoc($dataset);
            /* Якщо шаблон існує у базі даних, відкриваємо відповідний файл */
            $template=fopen($datarow["template_file"],"r");
            mysqli_free_result($dataset);
            if (!$template)
            {
                die("<h1><b>Error</b></h1> <hr> <h3>Template not found!<br>Cannot
show page</h3>");
            }
            /*  Якщо файл існує, зчитуємо його вміст для подальшого виводу*/
            while ($fs=fgets($template))
            {
               $pos=strpos(strtolower($fs),"</head>");
               if ($pos===false)
               {
                  $head=$head.$fs;
               }
               else
               {
                  for ($i=($pos+7); $i<strlen($fs); $i++)
                  {
                     $body=$body.$fs[$i];
                  }
                  break;
```

```
            }
        }
        while ($fs=fgets($template)) $body=$body.$fs;
    }
    …
?>
```

Але безпека системи полягає не лише у безпечному зберіганні інформації, але й у розмежуванні доступу до неї у рамках системи. У веб-сайтах значна увага приділяється розмежуванню доступу користувачів до різних веб-сторінок та їх елементів. Наприклад, блокування посилань для скачування файлів для незареєстрованих користувачів, а також можливість модератора видаляти коментарі будь яких користувачів сайту, при можливості звичайних відвідувачів оперувати лише з власними коментарями.

Для забезпечення розмежування доступу до сторінок веб-сайту під керуванням системи MyCMS була розроблена система, яка приймає рішення про відображення або приховування того чи іншого елементу веб-сторінки на етапі її збирання шаблонізатором, на основі наборів правил, розроблених користувачем.

Кожне правило набору являє собою трійку: умова, дія, параметр. Деякі дії не потребують параметрів, в такому разі останній елемент правила опускається. З правил формуються набори, які оброблюються системою зверху вниз. Якщо умова вказана у правилі приймає значення «істина», то виконується відповідна дія. Виконання хоча б одного з правил набору перериває його подальшу обробку. Набір правил також включає у себе дію за замовчуванням, яка виконується за невиконання усіх правил набору.

Для зберігання наборів правил у базі даних був розроблений простий формат їх запису. Набір правил починається конструкцією !S і закінчується конструкцією /!S. Кожне правило представляє собою конструкцію виду

!R ^c:~Умова~^a:~Дія~^p:~Параметр~ /!R

У системі розмежування доступу ядра MyCMS доступними є 4 різних дії:

Permit - відобразити елемент на сторінці;

Deny - не відображати елемент;

Msg - вивести замість елемента сторінки повідомлення;

Redir - переадресація на іншу веб-сторінку.

Дії “permit” та “deny” не потребують параметрів, а “msg” потребує в якості параметру рядок повідомлення, якщо від відсутній, то ця дія аналогічна “deny”, “redir” потребує URL цільової сторінки для переадресації.

Умова представляє собою SQL-запит до бази даних. Якщо запит повертає не порожню множину записів, то умова вважається виконаною. Дані про поточного користувача в системі передаються у запит-умову за допомогою змінних, які замінюються перед виконанням запиту на їхні фактичні значення. На данний момент визначено 3 змінні, ідентифікуючі користувача:

!var:userlogined! – приймає значення “true” якшо поточний користувач авторизований у системі, “false” - якщо ні.

!var:userid! – ідентифікатор поточного користувача;

!var:passmd5! – MD5-хеш пароля поточного користувача.

Приклад набору правил, який може використовуватися, наприклад, для розмежування доступу до елементів керування коментарями між звичайними користувачами та адміністрацією сайту(користувачі з рівень доступу для облікових записів яких більше або дорівнює 75). Дія за замовчуванням дозволяє переадресувати неавторизованого користувача на сторінку із формою для входу в систему.

```
!S
!R^c:~SELECT * FROM users WHERE
(user_id=!var:userid!)AND(user_accesslevel>=75) LIMIT 1~^a:~permit~/!R
!R^c:~SELECT * FROM users WHERE user_id=!var:userid! LIMIT 1~
^a:~deny~ /!R
!Default^a:~redir~ ^p:~www.mycms.ua/login.php~
/!S
```

2.3 Керування системою розмежування доступу до інформації

Керування системою розмежування доступу в розробленому програмному комплексі здійснюється за допомогою візуального редактора MyCMS. Редактор дозволяє під час редагування вмісту динамічних веб-сторінок прив’язувати до елементів сторінки наперед підготовані набори правил доступу. Приклад вікна редактора з прив’язкою набору правил на рис. 2.2.

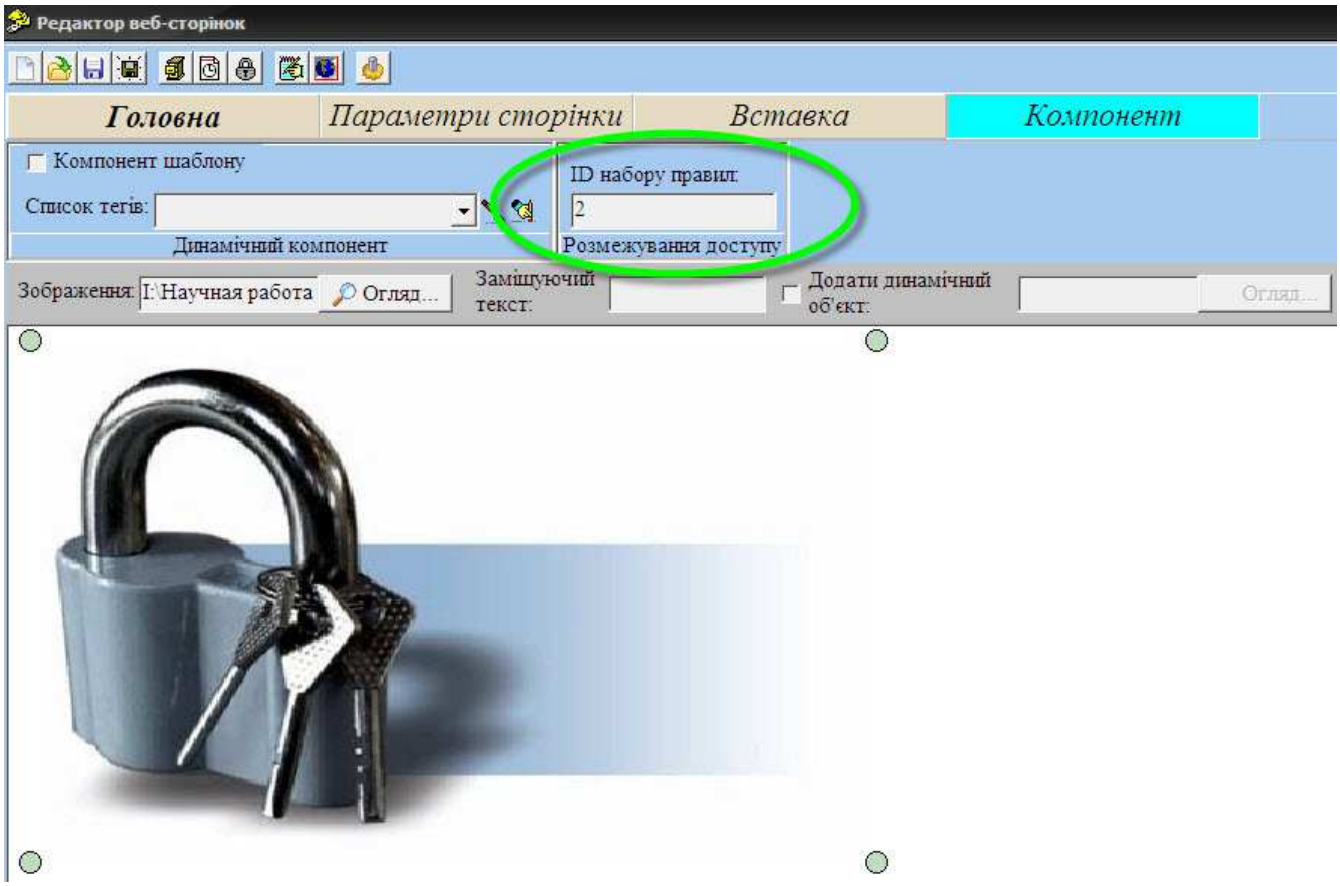

Рис. 2.2 Прив’язка набору правил доступу до елемента сторінки

Для зручної побудови наборів правил розмежування доступу для візуального редактора системи MyCMS засобами середовища програмування Borland Delphi 7 розроблений спеціальний модуль «Менеджер правил доступу» [9,10].

Даний модуль виконує більшу частину роботу з побудови наборів правил та запису їх до бази даних системи. Користувач за допомогою елементів інтерфейсу може створювати та видаляти набори правил і окремі правила всередині наборів, редагувати правила, змінювати порядок їх слідування у наборі. «Менеджер правил

доступу» викликається кнопкою  на панелі інструментів, розташованою над стрічкою. Приклад вікна модуля «Менеджер правил доступу» на рис. 2.3.

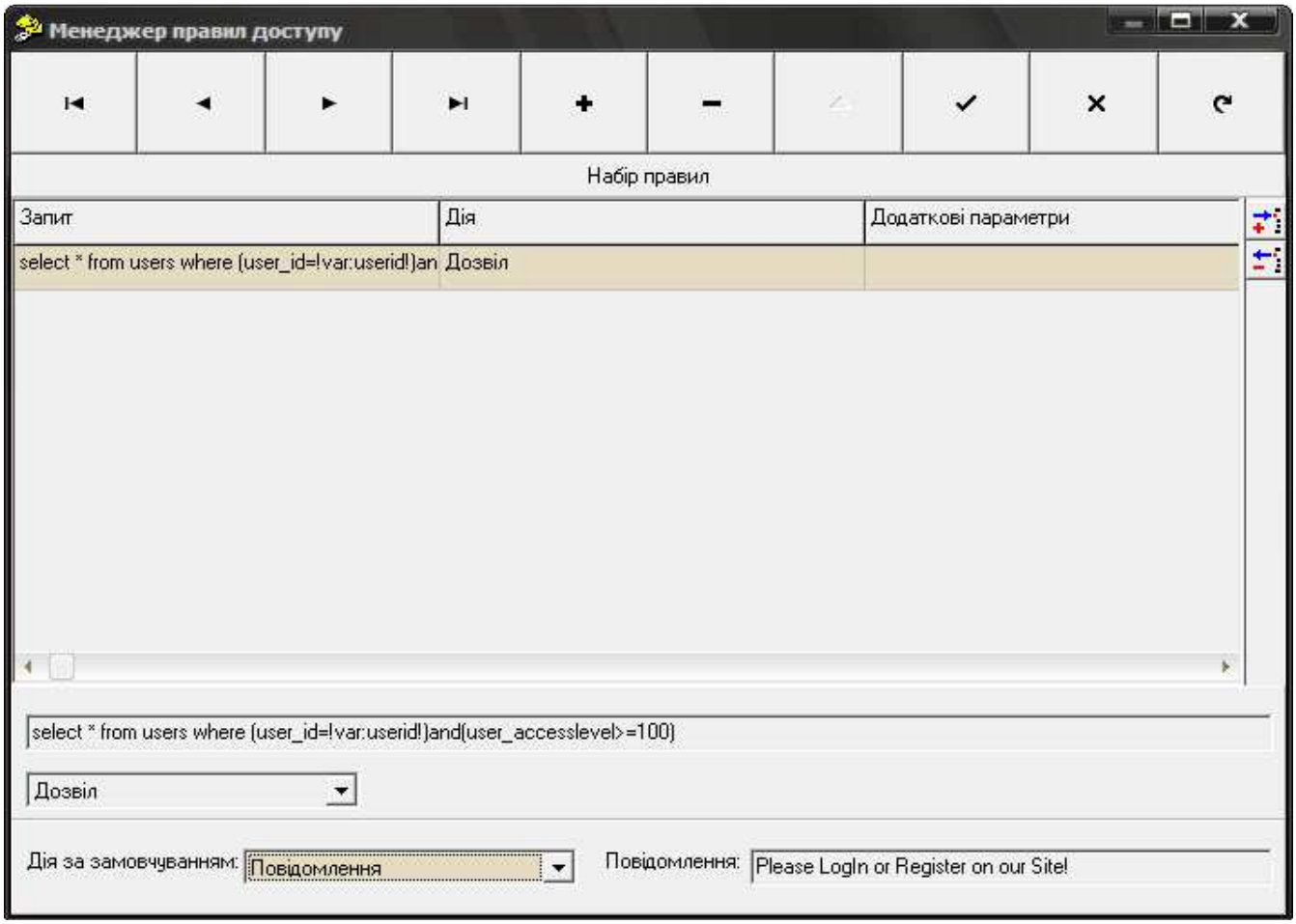

Рис.2.3 Вікно модуля «Менеджер правил доступу»

Зміни до набору правил у базі даних вносяться автоматично. Нижче наведений код процедури BuildSecurityScript, яка виконує кодування набору даних, створеного користувачем у вікні «Менеджера правил доступу», у спеціальний формат, в якому він зберігається у базі даних.

```
procedure BuildSecurityScript;
  var i:Longint;
      Script:AnsiString;
begin
  UnderConstruction:=True;
  with FSecurity do
  begin
```

```
    MRules.Clear;
    Script:='!s';
    for i:=1 to RulesGrid.RowCount-1 do
    begin
      Script:=Script+'!r^c:~'+RulesGrid.Cells[0,i]+'~';
      if RulesGrid.Cells[1,i]='Дозвіл' then
        Script:=Script+'^a:~permit~';
      if RulesGrid.Cells[1,i]='Заборона' then
        Script:=Script+'^a:~deny~';
      if RulesGrid.Cells[1,i]='Повідомлення' then
        Script:=Script+'^a:~msg~^p:~'+RulesGrid.Cells[2,i]+'~';
      if RulesGrid.Cells[1,i]='Преадресація' then
        Script:=Script+'^a:~redir~^p:~'+RulesGrid.Cells[2,i]+'~';
      if RulesGrid.Cells[1,i]='Зміна користувацької інформації' then
        Script:=Script+'^a:~set~^p:~'+RulesGrid.Cells[2,i]+'~';
      Script:=Script+'/!r';
    end;
    case CBDefaultAction.ItemIndex of
     0: Script:=Script+'!default^a:~permit~';
     1: Script:=Script+'!default^a:~deny~';
     2: Script:=Script+'!default^a:~msg~^p:~'+EExt.Text+'~';
     3: Script:=Script+'!default^a:~redir~^p:~'+EExt.Text+'~';
     4: Script:=Script+'!default^a:~set~^p:~'+EExt.Text+'~';
    end;
    Script:=Script+'/!s';
    MRules.Lines.Add(Script);
   end;
   UnderConstruction:=False;
  end;
```

Отже, захист від атак типу SQL-Injection забезпечується тим, що параметри, отримані від користувача не фігурують безпосередньо у запитах до бази даних.

ВИСНОВКИ

1. Захист інформації – першочергове завдання для програмного забезпечення, що керує роботою веб-сайту, тому при виконанні теоретичної частини дослідження були визначені основні структурні складові систем захисту інформації та їх функції:

- служба таємності даних;
- служба аутентифікації;
- служба цілосності даних;
- служба керування доступом;
- служба цілосності інформації;
- служба доставки.

2. Проаналізовані основні загрози безпеці веб-сайтів та визначені основні методи захисту:

- способи захисту, що потребують дій від користувача;
- способи, що не потребують дій від користувача;
- реєстрація користувачів та модерація коментарів;
- ретельна перевірка даних, отриманих від користувачів.

3. Проаналізовані спеціальні програми і програмні комплекси, призначені для захисту інформації в інформаційних системах: PGP, mod_security, mod_rewrite, CrackLib і визначені основні напрямки успішного захисту для власної розробки проти таких видів атак на веб-сайти, як Code injection, SQL Injection, XSS.

4. Розроблена стратегія захисту проти зазначених атак і реалізована у практичній частині дослідження.

5. В результаті виконання практичної частини дослідження розроблена система захисту інформації на веб-сайтах, створених за допомогою системи MyCMS, яку автор представляв у минулому році.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Спецвыпуск журнала «Хакер» № (2)75, февраль 2007 г. Издательский дом ООО «Гейм Лэнд»;
2. Закон України "Про електронні документи та електронний документообіг" від 22 травня 2003 р. № 851-IV;
3. Закон України "Про захист інформації в автоматизованих системах";
4. Закон України "Про цифровий підпис" від 22 травня 2003 р. № 852-IV;
5. http://captcha.opti-mail.net/ - Защита web-сервисов от спама. Ярослав Полещук
6. http://ru.wikipedia.org/wiki/CAPTCHA
7. http://ru.wikipedia.org/wiki/PHP-инъекция
8. Коггзолл, Джон. PHP 5. Полное руководство. : Пер. с англ.- М.: Издательский дом: «Вильямс», 2006. – 752 с. : ил. – Парал. тит. англ.
9. А. Я Архангельский. Програмирование в Delphi. Учебник по классическим версиям Delphi. – М.: ООО «Бином-Пресс», 2008 г. – 816 с.(1158 с.): ил.
10. Флёнов М.Е. Библия Delphi. – 2-е узд., пререраб. и доп. – СПб.: БХВ-Петербург, 2008 г. – 800 с.: ил.