ҒТАМР 81.93.29 https://doi.org/10.58805/kazutb.v.3.24-500

SQL-ИНЪЕКЦИЯЛАРЫНАН ВЕБ-ҚОСЫМШАЛАР ДЕРЕКТЕР ҚОРЫНЫҢ ОСАЛДЫҒЫН ЖӘНЕ ҚОРҒАНЫСЫН ЗЕРТТЕУ

Б. Рысбекқызы✉, Д.Т. Кожанова, Г.К. Кабиева, Ә.Ж. Изат, Л.Б. Кадырова, А.М. Леонтьева

Әбілқас Сағынов атындағы Қарағанды техникалық университеті, Қарағанды, Қазақстан

✉Корреспондент-автор: Bakhytgulz@mail.ru

Қазіргі уақытта веб-технологиялар интернет-дүкендерде, банктерде, кәсіпорындардың веб-парақшаларында барлық жерде қолданылады. Есесіне, деректер қоры көбінесе веб-қосымшаларды жазу үшін қолданылады. Деректер қоры ақпараттық қауіпсіздікке төнетін қауіптермен сипатталады. Деректерге қол жеткізу үшін аутентификация процесінде шабуылдаушы SQL Injection типті ақпараттық шабуылды қолдана алады. Бұл шабуылдың мәні веб-технологиялар мен SQL қиылысындағы қатені пайдалану болып табылады. Бұл пайдаланушы деректерін өңдеуге арналған көптеген веб-парақшалар зиянды кодты енгізуге әкелуі мүмкін деректер қорына арнайы SQL сұранысын қалыптастыратындығына байланысты.

Мақалада SQL инъекцияларынан веб-қосымшалар деректер қорының негізгі қауіптері мен осалдығы қарастырылады. Жұмыс SQL инъекцияларының негізгі түрлерін, осалдықтарды анықтау және алдын алу әдістерін және веб-қосымшалардың осалдығын тексеру тәсілдерін талдауды қамтиды. Деректер базасын SQL инъекцияларынан қорғау әдістеріне және тәжірибені дамытуға баса назар аударылады.

Зерттеу тақырыбы өзекті, өйткені заманауи әлемде интернет біздің өміріміздің ажырамас бөлігіне айналды. Қосымшаларды пайдаланудың өсуімен құпия деректерді қорғау үшін қауіпсіздік шараларына деген қажеттілік артты. Веб-қосымшалардың осалдығы қауіпсіздіктің бұзылуына әкелуі мүмкін, бұл пайдаланушылардың жеке және қаржылық ақпаратын жоғалтуға, ұйымдардың беделіне нұқсан келтіруге және заңды салдарға әкелуі мүмкін.

Түйін сөздер: SQL-инъекция, веб-қосымша, деректер қоры, деректерді қорғау, шабуыл, ақпараттық қауіпсіздік.

ИССЛЕДОВАНИЕ УЯЗВИМОСТИ И ЗАЩИТЫ БАЗ ДАННЫХ ВЕБ ПРИЛОЖЕНИЙ ОТ SQL-ИНЪЕКЦИЙ

Б. Рысбекқызы✉, Д.Т. Кожанова Г.К. Кабиева, А.Ж. Изат, Л.Б. Кадырова, А.М. Леонтьева

Карагандинский технический университет имени Абылкаса Сагинова, Караганда, Казахстан,

e-mail: Bakhytgulz@mail.ru

В настоящее время веб-технологии используются повсеместно в интернет-магазинах, банках, веб-страницах предприятий. При этом базы данных часто используются для написания веб-приложений. Для базы данных характерны угрозы информационной безопасности. В процессе аутентификации для получения доступа к данным злоумышленник может использовать информационную атаку типа SQL Injection. Суть данной атаки заключается в использовании ошибки на стыке веб-технологий

и SQL. Это обусловлено тем, что многие web-страницы для обработки пользовательских данных, формируют специальный SQL запрос к базам данных, что может привести к внедрению вредоносного кода.

В статье рассматриваются основные угрозы и уязвимость баз данных веб-приложений от SQL-инъекций. Работа включает в себя анализ основных видов SQL-инъекций, методов обнаружения и предотвращения уязвимостей, а также подходы к тестированию уязвимостей веб-приложений. Основное внимание уделено методам защиты баз данных от SQL-инъекций и разработке практических рекомендаций по обеспечению безопасности веб-приложений.

Тематика исследования актуальна так как в современном мире, так как интернет стал неотъемлемой частью нашей жизни. С ростом использования приложений также возросла потребность в мерах безопасности для защиты конфиденциальных данных. Уязвимости веб-приложений могут привести к нарушениям безопасности, что может привести к потере личной и финансовой информации пользователей, ущербу для репутации организаций и юридическим последствиям.

Ключевые слова: SQL-инъекция; веб-приложение; базы данных; защита данных; атака; информационная безопасность.

RESEARCH OF VULNERABILITY AND PROTECTION OF WEB APPLICATION DATABASES FROM SQL INJECTIONS

B. Rysbekyzy✉, D.T. Kozhanova G.K. Kabieva, A.Zh. Izat, L.B.Kadyrova, A.M.Leontyeva

Abylkas Saginov Karaganda Technical University, Karaganda, Kazakhstan,

e-mail: Bakhytgulz@mail.ru

Currently, web technologies are used everywhere in online stores, banks, and enterprise web pages. At the same time, databases are often used to write web applications. The database is characterized by information security threats. During the authentication process, an attacker can use an information attack such as SQL Injection to gain access to data. The essence of this attack is to exploit an error at the intersection of web technologies and SQL. This is due to the fact that many web pages form a special SQL query to databases to process user data, which can lead to the introduction of malicious code.

The article discusses the main threats and vulnerability of web application databases from SQL injections. The work includes an analysis of the main types of SQL injections, methods for detecting and preventing vulnerabilities, as well as approaches to testing web application vulnerabilities. The main focus is on methods for protecting databases from SQL injections and developing practical recommendations for ensuring the security of web applications.

The topic of the research is relevant in the modern world, since the Internet has become an integral part of our lives. As the use of apps has increased, the need for security measures to protect sensitive data has also increased. Web application vulnerabilities can lead to security breaches that can result in the loss of users' personal and financial information, damage to organizations' reputations, and legal consequences.

Keywords: SQL injection; web application; databases; data protection; attack; information security.

Кіріспе. Веб-қосымшадағы шабуылдардың ең танымал түрлерінің бірі SQL инъекциясы болып табылады. SQL-инъекция - бұл зиянды пайдаланушыға өзінің жеке параметрлерін қолданыстағы SQL-сұраныс жасауына немесе SQL-сұраныс жасауды болдырмауға және өзінің жеке сұраныс жа-

сауын ұсынуға мүмкіндік беретін SQL дерекқорына арнайы бағытталған инъекциялық шабуыл түрі.

SQL-инъекция ең кең таралған, бірақ жалғыз түрі емес. Инъекциялық шабуылдар екі негізгі компоненттен тұрады: интерпретатор және пайдаланушыдан түсетін пайдалы жүктеме, ол қандай да бір интерпретаторға оқылады [1-2]. Бұл инъекциялық шабуылдар FFMPEG (бейнекомпрессор) сияқты командалық жолдың утилитасына, сондай-ақ, деректер қорына (дәстүрлі жағдайда SQL-инъекцияға) қарсы болуы мүмкін дегенді білдіреді.

Материалдар мен әдістер. SQL-инъекция ең классикалық аталатын инъекция түрі болып табылады. SQL жолы HTTP пайдалы жүктемесінде экрандалады, бұл соңғы пайдаланушының атынан пайдаланушы SQL сұраныстарына әкеледі.

Дәстүрлі түрде көптеген OSS пакеттері PHP және SQL (жиі MySQL) комбинациясын пайдалана отырып жасалған. Әзірлеу тарихында-ғы SQL-инъекциялардың көптеген жиі айтылған осалдықтары PHP-нің түсініктер, логика және деректер коды арасындағы интерполяцияға неғұрлым жұмсақ тәсілді ұстануы нәтижесінде туындады. Ескі мектептің PHP әзірлеушілері өздерінің PHP файлдарында SQL, HTML және PHP комбинацияларын - PHP қолдайтын ұйымдастыру моделін қолданар еді, ол дұрыс қолданылмай, бұл осал PHP кодының көп болуына әкелетін еді.

Пайдаланушыға жүйеге кіруге мүмкіндік беретін форумның ескі бағдарламалық жасақтамасына арналған PHP код блогының мысалын қарастырайық:

```
<?php if ($_SERVER['REQUEST_METHOD'] != 'POST') {
  echo'
  <div class="row">
  <div class="small-12 columns">
  <form method="post" action="">
  <fieldset class="panel">
  <center>
  <h1>Sign In</h1><br>
  </center>
  <label>
  <Input type="text" id="username" name="username"
  placeholder="Username">
  </label>
  <label>
  <input type="password" id="password" name="password"
  placeholder="Password">
  </label>
  <center>
  <input type="submit" class="button" value="Sign In">
  </center>
  </fieldset>
  </form>
  </div>
  </div>';
  } else {
  // the user has already filled out the login form.
  // pull in database info from config.php
  $Servername = getenv('IP');
```

```
$Username = $mysqlUsername;
$Password = $mysqlPassword;
$Database = $mysqlDB;
$Dbport = $mysqlPort;
$database = new mysqli($servername, $username, $password,
 $database,$dbport);
if ($database->connect_error) {
echo ”ERROR: Failed to connect to MySQL”;
die;
}
$sql = ”SELECT userId, username, admin, moderator FROM users
 WHERE username = ’”.$_POST[’username’].”’ AND
 password = ’”.sha1($_POST[’password’]).”’;”;
$result = mysqli_query($database, $sql);
}
```

Көріп отырғаныңыздай, берілген кіру кодында PHP, SQL және HTML араласқан. Сонымен қатар, SQL-сұранысы сұраныс жолағын жасамас бұрын ешқандай тазартусыз сұрау параметрлерін біріктіру негізінде жасалады [3].

HTML, PHP және SQL кодтарының тоғысуы PHP негізіндегі веб-қосымшалар үшін SQL инъекциясының енуіне едәуір жеңілдетті. Тіпті WordPress сияқты кейбір ірі OSS PHP қосымшалары бұған дейін осындай жағдайдың құрбаны болған.

PHP-ден алынған қауіпсіздік сабақтары басқа тілдерде көрініс тапты және қазіргі заманғы веб-қосымшаларда SQL-ді енгізуге байланысты осалдықтарды табу әлдеқайда қиын. Дегенмен, бұл әлі де мүмкін және қауіпсіз бағдарламалаудың озық әдістерін қолданбайтын қосымшаларда жиі кездеседі.

Нәтижелер мен талқылау. ***Инъекцияға қарсы қорғаныстарды зерттеу.*** Әдетте әзірлеушінің CLI және пайдаланушы жіберген деректері бар автоматтандырудың кез келген түрін жазуына дұрыс назар аудармауының нәтижесінде бұл шабуылдар әлі де жиі кездеседі.

Инъекциялық шабуылдар сондай-ақ, жазықтықтың кең ауданын қамтиды. Инъекцияны CLI-ге немесе серверде жұмыс істейтін кез келген басқа оқшауланған интерпретаторға қарсы пайдалануға болады (ол ОЖ деңгейіне жеткенде, оның орнына ол команда инъекциясына айналады) [4-5]. Нәтижесінде, инъекция стиліндегі шабуылдардан қалай қорғанатынымыз-ды қарастырғанда, мұндай қорғау шараларын бірнеше санатқа бөлу оңай.

Ең алдымен, біз SQL-инъекция шабуылдарынан қорғаныстың бағалауымыз керек - инъекцияның ең кең тараған және белгілі түрі. SQL инъекцияларынан қорғану үшін не істей алатынымызды зерттегеннен кейін, бұл қорғаныстардың қайсысы инъекциялық шабуылдардың басқа түрлеріне қолданылатынын көруге болады. Ақыр соңында, біз инъекциядан қорғаудың бірнеше жалпы әдістерін бағалай аламыз, олар инъекцияның негізіндегі шабуылдардың қандай да бір жиынына тән емес.

SSQL-инъекциясының салдарын жұмсарту. SQL инъекциясы – инъекциялық шабуылдың ең көп таралған түрі және сол сияқты қорғанудың ең оңай түрлерінің бірі. Ол өте кең таралғандықтан, әрбір дерлік күрделі веб-қосымшаға әсер етуі мүмкін (SQL дерекқорларының таралуына байланысты) SQL инъекциясына қарсы көптеген жұмсарту және қарсы шаралар әзірленді.

Сонымен қатар, SQL инъекциялық шабуылдар SQL интерпретаторында орын алатындықтан, мұндай осалдықтарды анықтау өте қарапайым болуы мүмкін [6]. Тиісті анықтау және жұмсарту стратегиялары бар болса, веб-қосымшаның SQL инъекциялық шабуылына ұшырау мүмкіндігі өте төмен.

SQL инъекциясын анықтау. Кодтық базаңызды

SQL инъекциялық шабуылдардан қорғауға дайындау үшін алдымен SQL инъекциясы қабылдайтын форманы және сіздің кодтық базаңыздағы ең осал болуы мүмкін орындарды қарап шығуыңыз керек [7].

Көптеген заманауи веб-қосымшаларда SQL операциялары сервер жағындағы маршрутизация деңгейінен кейін орындалады. Бұл клиент туралы ештеңе қызықтырмайтынын білдіреді.

Мысалы, веб-қосымшаның код репозиторийінің файлдық құрылымы бар, ол келесідей:

/api

/routes

/utils

/analytics

/routes

/client

/pages

/scripts

/media

Біз клиентті іздеуді жіберіп алатынымызды білеміз, бірақ біз талдау бағытын қарастыруымыз керек, өйткені ол OSS-те салынған болса да, ол аналитика деректерін сақтау үшін қандай да бір дерекқорды пайдалануы мүмкін. Егер деректер құрылғылар мен сессиялар арасында сақталса, олар серверлік жадта, дискіде (журналдарда) немесе дерекқорда сақталады.

Серверде көптеген қолданбалар бірден көп дерекқорды пайдаланатынын білуіміз керек. Бұл қолданба, мысалы, SQL Server-ді және MySQL-ді пайдаланады дегенді білдіруі мүмкін. Сондықтан сервердегі іздеу кезінде SQL сұраныстарын бірнеше SQL тілінің іске асырылуынан табу үшін ортақ сұраныстарды пайдалану қажет.

Бұдан басқа, кейбір серверлік бағдарламалар доменге тән тілді (DSL) пайдаланады, ол біздің атымыздан SQL шақыруларын орындауы мүмкін, бірақ бұл шақырулар өңделмеген SQL шақыруына ұқсас құрылымданбайды.

Қолданыстағы кодтық базаны SQL-инъекцияның әлеуетті тәуекелдеріне дұрыс талдау жасау үшін, біз барлық алдыңғы DSL және SQL түрлерінің тізімін жасап, оны бір жерде сақтауымыз керек.

Егер біздің қолданба Node.js қолданбасы болып табылса және ол мыналарды қамтыса:

- SQL сервері – NodeMSSQL адаптері (npm) арқылы
- MySQL – mysql адаптері (npm) арқылы

онда екі SQL іске асыруынан SQL сұраныстарын таба алатын код базасында іздеулерді құрылымдауды қарастыруымыз керек.

Бақытымызға орай, Node.js-пен жеткізілетін модульдерді импорттау жүйесі бұл тапсырманы JavaScript тілінің қолданылу аймағымен бірге жеңілдетеді [8]. Егер SQL кітапханасы пермодуль негізінде импортталса, сұрауларды табу импортты іздеу сияқты оңай болады:

```
const sql = require('mssql')
// OR
const mysql = require('mysql');
```

Екінші жағынан, егер бұл кітапханалар жаһандық деңгейде жарияланса немесе ата-аналық класстан мұраға қалса, іздеу жұмыстары біршама күрделене түседі.

Node.js үшін жоғарыда аталған екі SQL адаптерінің екеуі де .query (x) шақыруымен аяқталатын синтаксисті қолданады, бірақ кейбір адаптерлер әріптік синтаксисті қолданады:

```
const sql = require('sql');
const getUserByUsername = function
                          (username){
const q = new sql();
q.select('*');
q.from('users');
q.where('username = ${username}');
q.then((res) => {
return 'username is : ${res}';
});
};
```

Дайындалған нұсқаулықтар. Жоғарыда айтылғандай, SQL-сұраныстар бұрын кең таралған және өте қауіпті. Бірақ көптеген жағдайларда олардан қорғану онша қиын емес.

SQL-ді іске асырудың көпшілігі қолдай бастаған әзірлемелердің бірі дайындалған операторлар болып табылады. Дайындалған нұсқаулықтар SQL-сұраныстарда пайдаланушы-лар ұсынатын деректерді пайдалану кезінде тәуекелді едәуір төмендетеді. Сонымен қатар, дайындалған нұсқаулықтар өте оңай зерттеледі және SQL-сұраныстарды түзетуді айтарлықтай жеңілдетеді [9].

Дайындалған нұсқаулықтар көбінесе инъекциядан қорғаудың «бірінші желісі» болып саналады. Дайындалған нұсқаулықтарды іске асыру оңай, Интернетте жақсы құжатталған және инъекциялық шабуылдарды тоқтату үшін өте тиімді.

Дайындалған нұсқаулықтар айнымалылар үшін толтырғыш мәндері бар сұранысты алдын ала құрастыру арқылы жұмыс істейді. Олар байланыстырушы айнымалылар ретінде белгілі, бірақ көбінесе жай толтырғыш айнымалылар деп аталады. Сұраныс құрастырылғаннан кейін толтырғыштар әзірлеуші ұсынған мәндермен ауыстырылады. Осы екі сатылы процестің нәтижесінде сұраныс мақсаты пайдаланушы жіберген кез келген деректер қаралмай тұрып белгіленеді.

Дәстүрлі SQL сұрауында пайдаланушы жіберген деректер (айнымалылар) және сұраныстың өзі дерекқорға жол ретінде бірге жіберіледі. Бұл дегеніміз, егер пайдаланушы деректері манипуляцияланса, бұл сұраныстың ниетін өзгерте алады.

Дайындалған нұсқаулықты пайдаланған кезде, сұраныстың мақсаты пайдаланушы жіберген деректер SQL интерпретаторына жіберілмес бұрын нақты анықталғандықтан, сұраныстың өзін өзгерту мүмкін емес. Бұл пайдаланушыларға арналған SELECT операциясын кез келген жолмен экрандауға және ЖОЮ операциясына түрлендіруге болмайтынын білдіреді. Егер пайдаланушы бастапқы сұранысты орындамаса және жаңасын бастаса, SELECT операциясынан кейін қосымша сұранысты орындау мүмкін емес. Дайындалған нұсқаулар SQL енгізу тәуекелдерінің көпшілігін жояды және барлық дерлік негізгі SQL дерекқорларымен қамтамасыз етіледі: MySQL, Oracle, PostgreSQL, Microsoft SQL Server және т.б.

Дәстүрлі SQL-сұраныстар мен дайындалған нұсқаулықтар арасындағы жалғыз маңызды ымыраласу өнімділік болып табылады [10-11]. Дерекқорға бір өтініштің орнына дерекқор дайындалған нұсқаулықты ұсынады, одан кейін компиляциядан кейін және сұранысты орындау кезінде енгізу үшін айнымалылар болады. Көптеген қосымшаларда бұл өнімділікті жоғалту минималды болады.

Синтаксистік тұрғыдан дайындалған нұсқаулықтар дерекқордан дерекқорға және адаптерден адаптерге ерекшеленеді.

MySQL-де дайындалған нұсқаулықтар өте қарапайым келеді:

```
 PREPARE q FROM 'SELECT name,
        barCode FROM products
        WHERE price <= ?';
SET @price = 12;
EXECUTE q USING @price;
DEALLOCATE PREPARE q;
```

Осы дайындалған нұсқаулықта біз бағасы төмен өнімдер үшін MySQL дерекқорынан сұраймыз (біз аты мен штрих кодын қайтарғымыз келеді)?.

Біріншіден, сұранысымызды q атауымен сақтау үшін PREPARE операторын қолданамыз. Бұл сұраныс құрылады және пайдалануға дайын болады. Әрі қарай, @price айнымалы мәнін 12 мәніне меншіктейміз. Бұл, мысалы, электрондық коммерция сайтына қарсы сүзгіден өткізетін пайдаланушылар орнатуы үшін жақсы айнымалы болар еді. Содан кейін біз ? толтыру үшін @price беретін сұрауды орындаймыз. Соңында, біз оны жадтан жою үшін q бойынша DEALLOCATE қолданамыз, осылайша оның атаулар кеңістігі басқа нәрселер үшін пайдаланылуы мүмкін.

Бұл қарапайым дайындалған нұсқаулықта q @price көмегімен орындалмас бұрын құрасты-

рылады. Егер @price 5-ке тең болса да; UPDATE users WHERE id = 123 SET balance =10000, қосымша сұрау деректермен құрастырылмағандықтан іске қосылмайды.

Бұл сұраудың әлдеқайда қауіпсіз нұсқасы:

```
'SELECT name, barcode from products WHERE price <= ' + price + ';'
```

Сіз анық көріп отырғаныңыздай, дайындалған нұсқаулықтарды алдын ала құрастыру SQL инъекцияларын жеңілдетудегі маңызды бірінші қадам болып табылады және оны веб-қосымшаңызда мүмкін болған жерде пайдалану керек [12].

Деректер қорына тән қорғаныстар. Кеңінен қабылданған дайындалған нұсқаулықтарға қосымша әрбір SQL негізгі дерекқоры қауіпсіздікті арттыру үшін өзінің жеке функцияларын ұсынады. Oracle, MySQL, MS SQL және SOQL SQL сұраныстарында пайдалану үшін қауіпті деп саналатын таңбалар мен таңбалар жиынтығын автоматты түрде экрандау әдістерін ұсынады. Осы санациялар туралы шешім қабылданатын әдіс нақты пайдаланылатын дерекқор мен механизмге байланысты болады.

Oracle (Java) келесі синтаксиспен шақырылуы мүмкін кодтаушыны ұсынады:

```
ESAPI.encoder().enodeForSQL(new OracleCodec(), str);
```

Осыған ұқсас, MySQL баламалы функционалдылықты ұсынады. MySQL бағдарламасында дұрыс экрандалмаған жолдарды пайдалануды болдырмау үшін мыналарды пайдалануға болады:

```
SELECT QUOTE('test''case');
```

MySQL жүйесіндегі QUOTE функциясы кері қиғаш сызықтарды, жалғыз тырнақшаларды немесе NULL мәндерін болдырмайды және бір тырнақшалы жолды дұрыс қайтарады.

MySQL mysql_real_escape_string () ұсынады. Бұл функция барлық алдыңғы қисық сызықтарды және жалғыз тырнақшаларды болдырмайды, сонымен қатар, қос тырнақшаларды ,\n және\r (linebreak) болдырмайды.

Қатерлі таңбалар жиынтығын айналып өту үшін дерекқорға тәуелді жолдарды тазалау құралдарын пайдалану жолдан айырмашылығы SQL-литералды жазуды қиындата отырып, SQL-инъекция тәуекелін төмендетеді. Егер параметрленуі мүмкін емес сұраныс орындалса, олар әрқашан пайдаланылуы тиіс - бірақ оларды жанжақты қорғау ретінде емес, жұмсарту ретінде қарау керек.

Жалпы инъекциялық қорғаныс. SQL инъекцияларынан қорғану мүмкіндігіне қоса, сіз сондайақ, сіздің қосымшаңыздың инъекцияның басқа аз таралған түрлерінен қорғалғанына көз жеткізуіңіз керек [13]. Жоғарыда айтылғандай, инъекциялық шабуылдар командалық жолдың немесе интерпретатордың утилитасының кез келген түріне қарсы болуы мүмкін.

Біз SQL инъекцияларынан басқа мақсаттарды іздеуге және инъекцияға күтпеген осалдықтың пайда болу қаупін азайту үшін барлық қолданбаның логикасында әдепкі кодтау әдістерін қолдануға тиіспіз.

Инъекцияларға арналған ықтимал мақсаттар. Біз бейне немесе суреттерді сығудың CLI интерфейстері әлеуетті енгізу мақсаты ретінде пайдаланылуы мүмкін сценарийді қарастырған болатынбыз. Бірақ енгізу FFMPEG сияқты командалық жолдың утилиттерімен шектелмейді. Ол мәтіндік енгізуді қабылдайтын және мәтінді

қандай да бір интерпретатордың көмегімен түсіндіретін немесе мәтінді қандай да бір командалар тізімімен салыстыратын скрипттің кез келген түріне қолданылады.

Әдетте, жүйеге енгізуді іздеу кезінде тәуекел дәрежесі жоғары мынадай объектілер пайдаланылады:

- Міндеттерді жоспарлаушылар
- Қысу/оңтайландыру кітапханалары
- Қашықтағы сақтық көшірме скрипттері
- Дерекқорлар
- Тіркеушілер
- Хосттың кез келген операциялық жүйесін шақыру
- Кез келген интерпретатор немесе компилятор

Сіздің веб-қосымшаңыздың компоненттерін енгізудің әлеуетті тәуекелі тұрғысынан бірінші рет ранжирлеу кезінде оларды жоғарыда келтірілген жоғары тәуекел объектілерінің тізімімен салыстырыңыз. Бұл сіздің зерттеуіңізге арналған бастапқы нүктелеріңіз болып табылады.

Тәуелділіктер де тәуекел болуы мүмкін, себебі көптеген тәуелділіктер өздерінің (суб) тәуелділіктерін әкеледі, олар көбінесе осы санаттардың біріне түсуі мүмкін.

Ең аз өкілеттілік принципі. Ең аз өкілеттілік принципі (көбінесе ең аз артықшылықтар принципі деп аталады) абстракцияның маңызды ережесі болып табылады, оны қауіпсіз веб-қосымшаларды құруға тырысу кезінде әрқашан пайдалану керек. Бұл принцип кез келген жүйеде жүйенің әрбір мүшесінің өз жұмысын орындау үшін қажетті ақпарат пен ресурстарға ғана қол жеткізуі керек екенін айтады.

Бағдарламалық қамтамасыз ету әлемінде мынадай принцип қолданылуы мүмкін: «бағдарламалық жүйедегі әрбір модуль осы модульдің дұрыс жұмыс істеуі үшін қажетті деректер мен функцияларға ғана қол жеткізуге тиіс».

Теорияда бұл қарапайым болып көрінеді, бірақ қажет болған жағдайда масштабты веб-қосымшаларда сирек қолданылады. Бұл принцип шын мәнінде маңыздырақ болады, өйткені қолданба күрделілігі бойынша масштабталады, өйткені күрделі қолданбадағы модульдер арасындағы өзара әрекеттесу күтпеген жанама әсерлер әкелуі мүмкін.

Веб-қосымшаңызбен біріктірілген және пайдаланушы профильдерінің фотосуреттерінің сақтық көшірмесін автоматты түрде жасайтын пәрмен жолы интерфейсін жасап жатқаныңызды елестетіп көріңіз [14]. Бұл CLI терминалдан (қолмен сақтық көшірме жасау) немесе веб-қосымшаңыз орнатылған бағдарламалау тілінде жазылған адаптер арқылы шақырылады. Егер бұл CLI ең аз өкілеттік қағидаты бойынша құрылған болса, онда CLI бұзылған болса да, қосымшаның қалған бөлігі бұзылмас еді. Екінші жағынан, әкімші ретінде орындалатын CLI алаяқтық инъекциялық шабуыл анықталған және жұмыс істеген жағдайда бүкіл қолданба серверін аша алады.

Ең аз өкілеттілік принципі әзірлеушілер үшін кедергі болып көрінуі мүмкін - қосымша жазбаларды, бірнеше кілттерді басқару және т.б. - бірақ осы қағидатты дұрыс іске асыру сіздің қосымшаңыз бұзылған жағдайда ұшырайтын тәуекелді шектейді.

Ақ тізімге қосу пәрмендері. Инъекция үшін ең үлкен қауіп - бұл клиент (пайдаланушы) орындау үшін серверге командалар жіберетін веб-қосымшадағы функционалдылық. Бұл нашар архитектуралық тәжірибе және оны кез келген жағдайда болдырмау керек [15].

Пайдаланушы таңдаған пәрмендер серверде кез келген контексте орындалуы керек болғанда, оларға әлеуетті нұқсан келтіруге немесе қолданбаның күйін өзгертуге (дұрыс пайдаланбаған жағдайда) мүмкіндік беретін қосымша қадамдар қажет болады. Пайдаланушы пәрмендерінің сервермен тікелей түсіндірілуіне мүмкіндік берудің орнына, пайдаланушыға қолжетімді пәрмендердің нақты анықталған ақ тізімін жасау керек. Бұл команда синтаксисіне (тәртібі, жиілігі, параметрлері) қосымша ретінде барлығын қара тізім форматында емес, ақ тізім форматында сақтай отырып, бірге пайдаланылуы тиіс. Келесі мысалды қарастырайық:

Келесі мысалды қарастырайық:

```
<div class="options">
<h2>Commands</h2>
<input type="text" id="command-list"/>
<button type="button" onclick="sendCommands()">Серверге пәрмендерді
                                                жіберу </button>
</div>
const cli = require('../util/cli');
/*
* Клиенттен пәрмендерді қабылдайды, оларды CLI-ге қарсы іске қосады.
*/
const postCommands = function(req, res) {
cli.run(req.body.commands);
};
```

Бұл жағдайда клиент CLI кітапханасы қолдайтын кез-келген командаларды серверге қарсы орындай алады. Бұл дегеніміз, CLI жұмыс уақыты мен толық көлемі Соңғы пайдаланушыға CLI қолдайтын командаларды беру арқылы қол жетімді, тіпті егер олар әзірлеуші қолдануға арналмаған болса да.

Неғұрлым түсініксіз жағдайда, барлық пәрмендерге әзірлеуші рұқсат берген болуы мүмкін, бірақ синтаксис, тәртіп және жиілік серверде CLI-ге қарсы күтпеген функционалдық (инъекция) жасау үшін біріктірілуі мүмкін. Тез және лас жұмсарту - бұл тек бірнеше командалардың ақ тізімі:

```
const cli = require('../util/cli');
const commands = [
'print',
'cut',
'copy',
'paste',
'refresh'
];
/*
/*
* Клиенттен пәрмендерді қабылдайды, егер олар ақ тізім массивінде
*пайда болса, оларды ТЕК CLI-ге қарсы іске қосады.
*/
const postCommands = function(req, res) {
const userCommands = req.body.commands;
userCommands.forEach((c) => {
if (!commands.includes(c)) { return res.sendStatus(400); }
});
cli.run(req.body.commands);
};
```

Бұл жылдам және лас жұмсарту командалардың ретіне немесе жиілігіне қатысты мәселелерді шеше алмауы мүмкін, бірақ бұл клиент немесе соңғы пайдаланушы пайдалануға арналмаған командаларды шақыруға жол бермейді [16]. Қара тізім қолданылмайды, өйткені қосымшалар уақыт өте келе дамиды. Қара тізімдер пайдаланушыға қажетсіз функционалдылық деңгейлерін беретін жаңа пәрмен қосылған жағдайда қауіпсіздік қаупі ретінде қарастырылады.

Пайдаланушы енгізуі қабылданып, CLI-ге жіберілген кезде, әрқашан қара тізім тәсілін емес, ақ тізім тәсілін таңдаңыз.

Қорытынды. Инъекциялық шабуылдар классикалық түрде деректер қорына, атап айтқанда SQL деректер қорына жатады. Бірақ дерекқорлар дұрыс жазылған кодсыз және конфигурациясыз инъекциялық шабуылдарға осал болса да, API соңғы нүктесі (немесе тәуелділік) өзара әрекеттесетін кез келген CLI инъекцияның құрбаны болуы мүмкін.

Негізгі SQL дерекқорлары SQL инъекциясының алдын алу шараларын ұсынады, бірақ SQL инъекциясы қолданбаның архитектурасымен және қате жазылған клиент-сервер кодымен әлі де мүмкін. Кодтық базаға ең аз өкілеттік принципін енгізу сіздің ұйымыңызға және веб-қосымшаңыздың инфрақұрылымына келтірілген зиянды азайту арқылы бұзылған жағдайда веб-қосымшаңызға көмектеседі. Бастапқыда қауіпсіздік принциптеріне негізделген қосымша ешқашан клиентке (пайдаланушыға) серверде орындалатын сұраныс немесе пәрмен беруге мүмкіндік бермейді.

Егер пайдаланушы енгізген деректерді сервер жағындағы операцияларға түрлендіру қажет болса, бұл операциялар жалпы функционалдылықтың бір бөлігі ғана қолжетімді болуы үшін және қауіпсіздікті тексерудің жауапты тобы қауіпсіз ретінде тексерген функционалдылық қана ақ тізімге енгізілуі тиіс.

Осы басқару элементтерін пайдалана отырып, веб-қосымша инъекция стиліндегі осалдыққа ие болу ықтималдығы әлдеқайда төмен болады.

SQL-инъекция шабуылдары кеңінен танымал және оларға дайын болғанымен, инъекция шабуылдары сервер API сұранысына жауап ретінде пайдаланатын кез келген CLI утилитасына қарсы болуы мүмкін.

SQL дерекқоры (жиі) инъекциядан жақсы қорғалған. Автоматтандыру белгілі SQL-инъекция шабуылдарын тестілеу үшін өте ыңғайлы, өйткені шабуыл әдісі өте жақсы құжатталған. SQL-инъекция істен шыққан жағдайда, кескін компрессорларын, резервтік көшіру утилиттерін және командалық жолдың басқа интерфейстерін әлеуетті мақсатты нысандар ретінде қарастырыңыз.

Әдебиеттер

1.Ge, X., Paige, R.F., Polack, F.A., Chivers, H. and Brooke, P.J. Agile Development of Secure Web Applications. Proceedings of the 6th International Conference on Web Engineering. Palo Alto. 2006, - P:305-312. DOI 10.1145/1145581.1145641

2. Norwawi, N.M. and Selamat, M.H. Secure E-Commerce Web Development Framework// Infor-mation Technology Journal. 2011, Vol.10 (4) - P.769-778. DOI 10.3923/itj.2011.769.778

3.McGraw, G. and Viega, J. Building Secure Software. In RTO/NATO Real-Time Intrusion Detection Symp. 2002.

4.Mouratidis, H., Jürjens, J. and Fox, J. Towards a Comprehensive Framework for Secure Systems Development //Advanced Information Systems Engineering. Springer, Berlin Heidelberg.- 2006.- P. 48-62. DOI 10.1007/11767138_5

5.Гафнер, В. В. Информационная безопасность.Учебное пособие. В 2 ч. Ч.1 / В. В. Гафнер. – Екатеринбург:Урал.гос.пед.ун-т,2009.- 155 с. ISBN 978-5-7186-0414-6

6.Булыгина О.В. Методические указания по выполнению расчетно-графической работы по дисциплине «Безопасность веб-приложений» [Электронный ресурс]: электронные методические указания для студентов, обучающихся по направлению 10.04.01 «Информационная безопасность» / Булыгина О.В. – Электрон. дан. – Смоленск: РИО филиала ФГБОУ ВО «НИУ «МЭИ» в г. Смоленске, 2019.- Дата обращения: 26.10.2023.

7. Афанасьев А.А., Веденьев Л.Т., Воронцов А.А., Газизова Э.Р. Аутентификация. Теория и практика обеспечения безопасного доступа к информационным ресурсам.-Учебное пособие.-Москва: Горячая линия-Телеком. -2012. -550 с. - ISBN 978-59912-0257-2

8.Прохорова, О.В. Информационная безопасность и защита информации [Электронный ресурс]: учебник/ Прохорова О.В.— Электрон. текстовые данные.— Самара: Самарский государственный архитектурно-строительный университет, ЭБС АСВ, 2014-113 с.— Режим доступа: http://www.iprbookshop.ru/43183.-ЭБС «IPRbooks»- Дата обращения: 26.10.2023.

9.Буренин, С.Н. Web-программирование и базы данных [Электронный ресурс] : учеб. практикум / С. Н. Буренин. - Москва: Моск. гуманит. ун-т. 2014. -120 с. https://edu.rsreu.ru/res/specialities/disc_edu_programs/80240-edu_program-file-Web.- Дата обращения: 10.11.2023.

10.Бьюли, А. Изучаем SQL. Учебное пособие [Текст] / А. Бьюли – Символ-Плюс. 2007. -312 с.: ил. – ISBN 978-5-93286-051-9.

11.Jan vom Brocke, Jan Mendling. Business Process Management Cases: Digital Innovation and Business Transformation in Practice - Springer, 2018.- 610 p. ISBN 978-3-319-86372-6

12. Jacobson, A. Cakula, S. Automated Learning Support System to Provide Sustainable Cooperation between Adult Education Institutions and Enterprises // Procedia Computer Science.- 2015.- Vol. 43.- P. 127-133. DOI 10.1016/j.procs.2014.12.017

13.Huoing, M. Truong. Integrating learning styles and adaptive e-learning system: Current developments, problems and opportunities //Computers in Human Behavior.- 2016.- Vol.55, Part B. -P:1185-1193. DOI 10.1016/j.chb.2015.02.014

14.Marcella La Rosa, Pnina Soffer. Business Process Management Workshops – Springer. 2012.-P.840. ISBN-13 978-3642362866

15.Philipp J. Pratt, Mari Z. Last A Guide to SQL - Course Technolog.// Cengage Learning.- 320 p. 2014. ISBN-13 978-1111527273

16.PostgreSQL vs MySQL // habr [Электронный ресурс]. URL: https://habr.com/company/mailru/blog/248845/ -Д ата обращения: 21.04.2023.

References

1.Ge, X., Paige, R.F., Polack, F.A., Chivers, H. and Brooke, P.J. Agile Development of Secure Web Applications. Proceedings of the 6th International Conference on Web Engineering. Palo Alto. 2006, -P:305-312. DOI 10.1145/1145581.1145641

2. Norwawi, N.M. and Selamat, M.H. Secure E-Commerce Web Development Framework// Infor-mation Technology Journal. 2011, Vol.10 (4) - P.769-778. DOI 10.3923/itj.2011.769.778

3.McGraw, G. and Viega, J. Building Secure Software. In RTO/NATO Real-Time Intrusion Detection Symp. 2002.

4.Mouratidis, H., Jürjens, J. and Fox, J. Towards a Comprehensive Framework for Secure Systems Development //Advanced Information Systems Engineering. Springer, Berlin Heidelberg.- 2006.- P. 48-62. DOI 10.1007/11767138_5

5.Gafner, V. V. Informacionnaja bezopasnost'.Uchebnoe posobie. V 2 ch. Ch.1 / V. V. Gafner. – Ekaterinburg:Ural.gos.ped.un-t,2009.- 155 s. ISBN 978-5-7186-0414-6

6.Bulygina O.V. Metodicheskie ukazanija po vypolneniju raschetno-graficheskoj raboty po discipline «Bezopasnost' veb-prilozhenij» [Jelektronnyj resurs]: jelektronnye metodicheskie ukazanija dlja studentov, obuchajushhihsja po napravleniju 10.04.01 «Informacionnaja bezopasnost'» / Bulygina O.V. – Jelektron. dan. – Smolensk: RIO filiala FGBOU VO «NIU «MJeI» v g. Smolenske, 2019.- Data obrashhenija: 26.10.2023.

7. Afanas'ev A.A., Veden'ev L.T., Voroncov A.A., Gazizova Je.R. Autentifikacija. Teorija i praktika obespechenija bezopasnogo dostupa k informacionnym resursam.-Uchebnoe posobie.-Moskva: Gorjachaja linija-Telekom. -2012. -550 s. - ISBN 978-59912-0257-2

8.Prohorova, O.V. Informacionnaja bezopasnost' i zashhita informacii [Jelektronnyj resurs]: uchebnik/ Prohorova O.V.— Jelektron. tekstovye dannye.— Samara: Samarskij gosudarstvennyj arhitekturno-stroitel' nyj universitet, JeBS ASV, 2014-113 c.— Rezhim dostupa: http://www.iprbookshop.ru/43183.-JeBS «IPRbooks»- Data obrashhenija: 26.10.2023.

9.Burenin, S.N. Web-programmirovanie i bazy dannyh [Jelektronnyj resurs] : ucheb. praktikum / S. N. Burenin. - Moskva: Mosk. gumanit. un-t. 2014. -120 s.https://edu.rsreu.ru/res/specialities/disc_edu _programs/80240-edu_program-file-Web.- Data obrashhenija: 10.11.2023.

10.B'juli, A. Izuchaem SQL. Uchebnoe posobie [Tekst] / A. B'juli – Simvol-Pljus. 2007. -312 s.: il. – ISBN 978-5-93286-051-9.

11.Jan vom Brocke, Jan Mendling. Business Process Management Cases: Digital Innovation and Business Transformation in Practice - Springer, 2018.- 610 p. ISBN 978-3-319-86372-6

12. Jacobson, A. Cakula, S. Automated Learning Support System to Provide Sustainable Cooperation between Adult Education Institutions and Enterprises // Procedia Computer Science.- 2015.- Vol. 43.- P. 127-133. DOI 10.1016/j.procs.2014.12.017

13.Huoing, M. Truong. Integrating learning styles and adaptive e-learning system: Current developments, problems and opportunities //Computers in Human Behavior.- 2016.- Vol.55, Part B. -P:1185-1193. DOI 10.1016/j.chb.2015.02.014

14.Marcella La Rosa, Pnina Soffer. Business Process Management Workshops – Springer. 2012.-P.840. ISBN-13 978-3642362866

15.Philipp J. Pratt, Mari Z. Last A Guide to SQL - Course Technolog.// Cengage Learning.- 320 p. 2014. ISBN-13 978-1111527273

16.PostgreSQL vs MySQL//habr [Jelektronnyj resurs]. URL: https://habr.com/company/mailru/blog/ 248845/ -Data obrashhenija: 21.04.2023.

Авторлар туралы мәліметтер

Рысбекқызы Б. - PhD докторы, аға оқытушысы, Әбілқас Сағынов атындағы Қарағанды техникалық университеті, Қарағанды, Қазақстан, e-mail: Bakhytgulz@mail.ru;

Кожанова Д.Т. - оқытушысы, Әбілқас Сағынов атындағы Қарағанды техникалық университеті, Қарағанды, Қазақстан, e-mail: kadirova_dinara@mail.ru;

Кабиева Г.К. - оқытушысы, Әбілқас Сағынов атындағы Қарағанды техникалық университеті, Қарағанды, Қазақстан, e-mail: gulnarakabieva@mail.ru;

Изат А.Ж. - КазМИРР инженері, Әбілқас Сағынов атындағы Қарағанды техникалық университеті, Қарағанды, Қазақстан, e-mail: celovek.dobra7@gmail.com;

Кадырова Л.Б. - аға оқытушысы, Әбілқас Сағынов атындағы Қарағанды техникалық университеті, Қарағанды, Қазақстан, e-mail: lyakaeldos@mail.ru;

Леонтьева А.М. - аға оқытушысы, Әбілқас Сағынов атындағы Қарағанды техникалық университеті, Қарағанды, Қазақстан, e-mail: ternast@mail.ru.

Information about authors

Rysbekkyzy B. - PhD, senior lecturer, Abylkas Saginov Karaganda Technical University, Karaganda, Kazakhstan, e-mail: Bakhytgulz@mail.ru;

Kozhanova D.T. - lecturer, Abylkas Saginov Karaganda Technical University, Karaganda, Kazakhstan, e-mail: kadirova_dinara@mail.ru;

Kabieva G.K. - lecturer, Abylkas Saginov Karaganda Technical University, Karaganda, Kazakhstan, e-mail: gulnarakabieva@mail.ru;

Izat A.Zh. - engineer of KazMIRR, Abylkas Saginov Karaganda Technical University, Karaganda, Kazakhstan e-mail: celovek.dobra7@gmail.com;

Kadyrova L.B. - senior lecturer, Abylkas Saginov Karaganda Technical University, Karaganda, Kazakhstan, e-mail: lyakaeldos@mail.ru;

Leontyeva A.M. - senior lecturer, Abylkas Saginov Karaganda Technical University, Karaganda, Kazakhstan, e-mail: ternast@mail.ru.