DOI: 10.18372/2410-7840.19.11571
УДК 004.056.52

ЧАСОВА СКЛАДНІСТЬ РІЗНИХ РЕАЛІЗАЦІЙ ДИСПЕТЧЕРА ДОСТУПУ МЕРЕЖЕВОГО РЕСУРСУ

Денис Самойленко

Для створення комплексної системи захисту мережевого інформаційного ресурсу, у відповідності до нормативних вимог, необхідно реалізовувати концепцію диспетчера доступу як єдиної точки проходження усіх запитів. Існує декілька різних підходів реалізації диспетчера доступу, що відрізняються показниками безпеки щодо надійності. Популярною є думка, що найбільш якісний з точки зору інформаційної безпеки варіант є одночасно й найповільнішими, що погіршує якість послуги безпеки «доступність». Ґрунтовних досліджень балансу показників надійності та доступності по відношенню до диспетчерів доступу раніше не проводилось. Реалізовано чотири варіанти диспетчера доступу засобами сервера Apache та мови програмування PHP; здійснено експериментальне визначення їх часової складності за допомогою апарата регресійного аналізу. Показано, що відмінності у часовій складності не перевищують 5 %, що дозволяє надати перевагу найнадійнішому варіантові без погіршення доступності. Додатково засвідчено кращу часову ефективність мови PHP 7 у порівнянні з PHP 5 при реалізації клієнт-серверної взаємодії. Використання одержаних результатів дозволить покращити показники якості безпеки мережевих інформаційних ресурсів.

Ключові слова: *кібербезпека, мережевий ресурс, диспетчер доступу, Apache, PHP.*

Стрімка еволюція інформаційних технологій, розвиток та глобалізація інформаційного суспільства супроводжуються зростанням загроз інформаційній безпеці. Питання кібербезпеки в усьому світі виходить на перші щаблі як всередині держав, так і на міждержавному рівні. Пріоритетність та необхідність покращення інформаційної безпеки на національному рівні встановлюється Конституцією України (ст. 17), Доктриною інформаційної безпеки України, Стратегією кібербезпеки України, Стратегією розвитку інформаційного суспільства в Україні та багатьма іншими законами та нормативними документами. Однією з задач інформаційної безпеки є захист мережевих інформаційних ресурсів (МІР) – сайтів, порталів, електронних бібліотек та архівів, банків, тощо.

При проектуванні захищених МІР з комплексною системою захисту інформації необхідно дотримуватись ряду вимог нормативних документів. Однією з таких вимог є створення диспетчера доступу (ДД) – єдиної точки проходження усіх звернень до МІР, як до інформаційної системи [2, 5].

Визначальною відмінністю МІР від автоматизованих систем класу «1» (одномашинні однокористувачеві комплекси) [5] з точки зору проходження запитів від користувачів полягає у багатоетапному процесі передачі надхідної інформації до кінцевого обробника (скрипта МІР). Схематично, такий процес можна уявити послідовністю трьох кроків: встановлення з'єднання («рукостискання»), оброблення запиту серверним програмним забезпеченням (ПЗ), запуск кодів МІР з передачею результатів попередньої обробки або, у разі неспроможності оброблення, формування повідомлення щодо помилки. Відсутність можливості впливу на роботу серверного ПЗ засобами МІР є головною проблемою, що ускладнює проектування ДД.

Щодо покращення показників безпеки клієнт-серверної взаємодії можна зазначити декілька типових підходів. Одним з найбільш поширених є встановлення міжмережевого екрану (Firewall, брандмауер), що контролює перші етапи оброблення запитів до моменту їх передачі до МІР. У роботі [1] зазначений підхід для безпеки веб-додатків названо універсальним та найбільш оптимальним.

Другим підходом є взагалі повна заміна механізмів клієнт-серверної взаємодії на альтернативні, що дозволяють контроль усіх етапів руху інформацій. Подібний підхід, зокрема, описаний у [4], де використовуються базові системні засоби керування підключеннями (сокетами) – Winsock.

Альтернативним підходом можна вважати «надбудови» над стандартними протоколами, що дозволяють покращити показники їх захищеності [3]. Як правило, додаткова функціональність полягає у впровадженні криптографічних алгоритмів та протоколів, що дозволяють перетворювати дані та створювати «тунелі», на зразок віртуальних приватних мереж (VPN).

Наведені підходи не дають вичерпний перелік захисних заходів, проте, відбивають базові ідеї, що в них закладаються. При цьому можна зауважити щодо їх головного недоліку – вони вимагають втручання у роботу сервера через встановлення додаткових пристроїв чи програмного забезпечення.

У більшості реальних випадків МІР розміщується на умовах хостингу на сервері, що не є власністю розробника МІР, зі значними обмеженнями

щодо встановлення пристроїв чи програм. Фактично, для встановленого МІР дозволено стандартне керування завершальними етапами роботи серверного ПЗ – інструкціями файлів «.htaccess», «robots.txt», тощо.

За таких умов, найбільш поширеним способом корегування роботи серверного ПЗ є застосування механізму фільтрації запитів за встановленими правилами (RewriteRule) з серверного модуля mod_rewrite. Основні висновки, одержані у попередніх роботах [6], дозволили сформулювати вимоги для коректного складання ДД. Основною з вимог є максимальна відмова від прийняття рішень на етапі серверного оброблення і якнайшвидша передача усіх запитів до МІР.

Як правило, первинний аналіз запитів користувачів здійснюється за принципом «білого переліку» - прямої відповідності запиту заданій структурі. Зазначений аналіз може бути здійснений як засобами серверного ПЗ (mod_rewrite), так і засобами самого МІР (мовою програмування).

Популярною є думка [напр. 7], що інструкції mod_rewrite є швидшими і саме ним має бути надана перевага. У противагу до неї, можна зазначити, що робота модуля mod_rewrite базується на регулярних виразах (Regular Expressions), оброблення яких є значно складнішим за інструкції порівняння у мовах програмування. У той же час, мова програмування (наприклад, PHP) є таким саме модулем сервера, як і mod_rewrite, тобто швидкодія інструкцій PHP не повинна поступатись інструкціям інших модулів. Ґрунтовних досліджень з порівняльного аналізу швидкодії різних реалізацій ДД у вільному доступі виявлено не було.

Метою даної роботи є реалізація диспетчерів доступу мережевих інформаційних ресурсів, побудованих на різних принципах аналізу «білого переліку», та проведення порівняльного аналізу їх швидкодії, як визначальної характеристики часової складності.

Експеримент. У якості предмета дослідження було складено ДД, що реалізовує перевірку трьох «білих записів»: головна сторінка (порожній запит - ^$), запит авторизації (/login), множинний запит (/news/номер), де номер – деяке число. За умови збігу запиту з шаблоном здійснюється перехід на відповідний обробник запиту у окремому файлі (index1-3.php). Інші запити спрямовуються на обробник помилок (error.php). Правила оброблення виглядають наступним чином:

```
RewriteRule ^$ index1.php [L]
RewriteRule ^login$ index2.php [L]
RewriteRule ^news/(\d+)$ index3.php?n=$1 [L,QSA]
RewriteRule .* error.php.
```

Опція [L] упереджує оброблення подальших правил, [QSA] – зберігає початкові параметри запиту, якщо такі були.

Окрему увагу слід зосередити на реалізації останньої інструкції зі спрямування запитів, що не пройшли фільтри, до обробника помилок. Оскільки сервер використовує серію перевірок запитів (повторний аналіз правил), усі запити врешті решт будуть спрямовуватись до обробника помилок. Наприклад, порожній запит після початкового аналізу обробиться першим правилом і перетвориться на «index1.php». Після повторного аналізу він не відповідатиме жодному з правил і буде замінений на «error.php». Після третього аналізу запит не зміниться, що означатиме припинення циклу аналізів.

Упередження повторного аналізу правил можна реалізувати двома способами: 1) контролем значення серверної змінної REDIRECT_STATUS, 2) встановленням опції [END] для інструкцій RewriteRule. Останній спосіб придатний для серверів Apache починаючи з версії 2.3, перший є універсальним.

Для аналізу було реалізовано обидва варіанти зупинки повторного оброблення. Повний склад файлів «.htaccess», що реалізують зазначені вище варіанти ДД наведено нижче.

```
#Варіант 1:
RewriteEngine On
RewriteRule ^$ index1.php [END]
RewriteRule ^login$ index2.php [END]
RewriteRule ^news/(\d+)$ index3.php?n=$1 [END,QSA]
RewriteRule .* error.php.
#Варіант 2:
RewriteEngine On
RewriteCond %{ENV:REDIRECT_STATUS} 200
RewriteRule ^ - [L]
RewriteRule ^$ index1.php [L]
RewriteRule ^login$ index2.php [L]
RewriteRule ^news/(\d+)$ index3.php?n=$1 [L,QSA]
RewriteRule .* error.php.
```

Для перенесення процесу аналізу запитів до самого МІР необхідно ретранслювати усі запити до єдиного файлу ДД «main.php». Відмова від використання для ДД стандартного файлу «index.php» здійснена з метою можливості контролю роботи ДД. Для практичного випробування було складено дві варіації файлу «.htaccess». У другій варіації додано опцію [END] для упередження повторного запуску аналізу через його потенційний вплив на загальну швидкодію ДД.

#Варіант 3:

```
RewriteEngine On
RewriteRule .* main.php.
#Варіант 4:
RewriteEngine On
RewriteRule .* main.php [END].
```

Оскільки час одного звернення до серверу є досить малим, для вимірювання використовувалось багаторазове послідовне звернення. З метою упередження буферизації відповіді, усі файли-обробники у якості відповіді генерували випадкові числа. Задля елімінування впливу на результати відмінностей у кодах МІР з різними ДД, у файлі main.php була повністю відновлена функціональність індексних файлів:

```
<?php
$request = $_SERVER['REQUEST_URI'];
if($request == '/'){
    echo '1 '.rand();
    exit();
}
if($request == '/login'){
    echo '2 '.rand();
    exit();
}
if(strpos($request,'/news/')===0){
    $n = substr($request,6);
    echo '3 ',$n,' ',rand();
    exit();
}
echo 'error<br>',rand().
```

Задля нівелювання впливу на роботу програм різних апаратних та програмних конфігурацій, як клієнт, так і сервер були розгорнуті на одному ПК. Використовувалась операційна система Windows (32 біт) із серверним ПЗ Apache 2.4.25. Спочатку, на клієнтському боці було встановлено PHP 7.1.1 (далі у тексті – PHP 7), на серверному – PHP 5.6.30 (PHP 5). У подальшому ролі клієнта та сервера були змінені.

Вимірювання. Як вже зазначалось вище, безпосереднє вимірювання часу відповіді сервера не можна вважати надійним підходом, в силу його малого значення. З метою максимального усунення похибок вимірювань застосовувався наступний підхід.

1. Встановлювався час роботи циклу з N=100 звернень до сервера.
2. Повторювався п. 1 для N=200-1000 з повним перезапуском циклу.
3. Дані пп. 1-3 оброблялись регресійним аналізом.
4. Пп. 1-3 виконувались окремо для дозволених запитів (усіх видів) та помилок.

Крок 2 дозволяє додатково (до впровадження випадкових чисел у відповідях) усунути наслідки буферизації відповідей сервера та нівелювати вплив на результати відмінності у програмних кодах, що реалізують продовження чи повторний запуск циклів.

Регресійний аналіз (п. 3) дозволяє виявити саме динамічну складову у зростанні часу відповіді, пов'язану зі збільшенням кількості запитів, ігноруючи постійну складову, спряжену з командами обслуговування циклів.

З метою зменшення складності підготовчих процедур, запит до сервера формувався безпосередньо через файловий сокет-клієнт (stream_socket_client), на відміну від більш поширеного використання складної бібліотеки сервер-серверної взаємодії с URL. Вимірювання часу здійснювалось прецизійною функцією «microtime», що визначає системний час з точністю до мілісекунд.

Фрагмент програми, що реалізовує пп. 1-2 алгоритму, наведено нижче

```
for($j=1;$j<11;$j++):
$t1 = microtime(true);
for($i=0;$i<100*$j;$i++):
$ssc = stream_socket_client ("127.0.0.1:81", $ern,$ers);
if(!$ssc){echo $ers;exit();}
fwrite($ssc,"GET /news/21 HTTP/1.1\r\nHost:localhost\r\nConnection:close\r\n\r\n");
$ans = "";
while($str = fgets($ssc))
$ans.= $str."<br>";
endfor;
$t2 = microtime(true);
echo $j*100, ' ',$t2-$t1,' <br>';
endfor.
```

З метою наближення вимірювань до умов реальної роботи МІР у склад алгоритму включено перевірку з'єднання на успішність, а також процес зчитування та збереження (у змінну) повної відповіді сервера. Саме такі дії виконуються у більшості реальних задач клієнт-серверної взаємодії.

Результати. Результати вимірювання і оброблення даних зведені до табл. 1-2. У табл. 1 дані відносяться до ситуації, коли клієнт використовує мову PHP 7, сервер – PHP 5. У табл. 2 – до зворотної ситуації. Числові дані визначають час одного сеансу взаємодії (запит від клієнта – відповідь сервера – одержання і зберігання відповіді клієнтом) у мілісекундах.

Таблиця 1

Час одного сеансу клієнт-серверної взаємодії, мс (PHP 7 - PHP 5)

	/	/login	/news/21	error	середнє	похибка
Варіант 1	9,45	7,92	8,56	7,84	8,44	0,13
Варіант 2	7,88	8,49	8,65	9,06	8,52	0,20
Варіант 3	7,83	8,31	8,22	8,47	8,21	0,36
Варіант 4	8,77	8,53	7,83	8,18	8,33	0,21

Таблиця 2

Час одного сеансу клієнт-серверної взаємодії, мс (PHP 5 - PHP 7)

	/	/login	/news/21	error	середнє	похибка
Варіант 1	9,93	11,05	11,09	10,22	10,57	0,17
Варіант 2	11,26	10,85	10,05	11,12	10,82	0,13
Варіант 3	10,96	10,13	9,07	10,81	10,24	0,42
Варіант 4	10,67	11,30	10,84	10,52	10,83	0,26

Обговорення. Як неважко помітити з наведених результатів, суттєвої відмінності між способами реалізації ДД не спостерігається. З урахуванням похибки вимірювань можна стверджувати про приблизну часову еквівалентність усіх ДД (у межах 5 %).

Тим не менш, найменший час сеансу демонструє ДД, реалізований засобами мови PHP. Одержаний висновок суперечить відзначеному вище популярному твердженню щодо повільності PHP [7], що додатково свідчить про відсутність попередніх досліджень реальних ситуацій з використанням для його формування лише експертних думок. Додатково можна зауважити, що побудова ДД засобами самого MIP є кращою з точки зору більшої локалізації захисних засобів всупереч їх розподіленості у інших ДД.

Звертає на себе увагу той факт, що використання опції примусової зупинки [END] не покращує швидкодію ДД, реалізованого засобами PHP (рядки 3-4 табл. 1-2). Це може бути пояснене тим, що процедура оброблення даної опції є більш складною, ніж повторне оброблення одного рядка ДД. Для більш складного ДД з декількома умовами зазначена опція призводить до прискорення роботи (рядки 1-2 таблиць).

Найбільшу відмінність демонструє вибір версії мови програмування для клієнта та сервера. При встановлені на клієнті PHP 7, а на сервері PHP 5 (табл. 1) середній час одного сеансу становить 8,38 мс, тоді як при зворотній комбінації – 10,62 мс (+27 %). Пояснення даному факту може бути дано через еволюційні покращення роботи з пам'яттю у PHP 7. При багаторазових підключеннях та зберіганнях відповіді у пам'яті клієнта механізм PHP 7 демонструє більш швидку роботу.

Висновки. Проведено вимірювання часової складності чотирьох варіантів реалізації диспетчера доступу мережевих інформаційних ресурсів. Показано, що принципових відмінностей у швидкодії різних ДД не спостерігається, відмінності у часі роботи не перевищують 5 %. Перевагу рекомендується надавати ДД, побудованому на програмному аналізі запитів засобами мови PHP. Додатково засвідчено на 27 % більшу швидкодію версії PHP 7 (у порівнянні з PHP 5) для задач оброблення серверних відповідей.

ЛІТЕРАТУРА

[1]. І. Василенко, "Універсальній метод захисту веб-додатків", *Системи обробки інформації*, № 1 (138). С. 122-124, 2016. [Електронний ресурс]. Режим доступу: http://www.hups.mil.gov.ua/periodic-app/article/15259/soi_2016_1_27.pdf [Дата доступу: травень 2017].

[2]. *НД ТЗІ 2.5-010-2003. Вимоги до захисту інформації WEB - сторінки від несанкціонованого доступу*, 20 с., 2003. [Електронний ресурс]. Режим доступу: http://dstszi.kmu.gov.ua/dstszi/doccatalog/document?id=106344 [Дата доступу: травень 2017].

[3]. В. Бурячок, В. Толубко, В. Хорошко, С. Толюпа, *Інформаційна та кібербезпека: соціотехнічний аспект: підручник*. К.: ДУТ, 288 с., 2015. [Електронний ресурс]. Режим доступу: http://www.dut. edu.ua/uploads/p_303_79299367.pdf [Дата доступу: травень 2017].

[4]. *Комплекс засобів захисту Web-ресурсів від несанкціонованого доступу «Тайфун-Web»*. Київ: Інститут комп'ютерних технологій, 11 с., 2014. [Електронний ресурс]. Режим доступу: http://www.ict. com. ua/ files/ websec/ Tfn_ Web_ OP_ r4.pdf [Дата доступу: травень 2017].

[5]. НД ТЗІ 2.5-004-99. Критерії оцінки захищеності інформації в комп'ютерних системах від несанкціонованого доступу, 61 с., 1999. [Електронний ресурс]. Режим доступу: http://dstszi.kmu.gov.ua/dstszi/doccatalog/document?id=106342 [Дата доступу: травень 2017].

[6]. Д. Самойленко, С. Данильченко, "Аналіз способів створення людино-орієнтованих імен з позиції інформаційної безпеки", *Матеріали V Всеукраїнської науково-практичної конференції з міжнародною участю «Сучасні проблеми інформаційної безпеки на транспорті»*. Миколаїв: Вид-во НУК, С. 108-109, 2015.

[7]. "Faster Option: Redirect via PHP to PHP or Apache mod rewrite redirect to PHP?" [Електронний ресурс]. Режим доступу: https://serverfault.com/ questions/ 88919/ faster- option- redirect- via-php-to-php-or-apache-mod-rewrite-redirect-to-php [Дата доступу: травень 2017].

REFERENCES

[1]. I. Vasilenko, "A universal method of web application protection", *Information Processing Systems,* no. 1(138), pp. 122-124, 2016.

[2]. *UKRAINE Security Service of Ukraine 2.5-010-2003. Requirements for information security of WEB-page from unauthorized access*. Kyiv: SSU, 2003.

[3]. V. Buryachok, V. Tolubko, V. Khoroshko, S. Tolyupa, *Information and cybersecurity: social and technical aspects,* 2015. State University of Telecommunications. [Online]. Available: http://www. dut.edu.ua/uploads/p_303_79299367.pdf [Accessed: May-2017].

[4]. *Web resource protection complex from unauthorized access*, 2014. Kyiv Institute of Computer Technologies. [Online]. Available: http://www. ict.com.ua/files/ websec/Tfn_Web_OP_r4.pdf [Accessed: May-2017].

[5]. *UKRAINE Security Service of Ukraine 2.5-004-99 Criteria for data protection from unauthorized access estimation in computer systems*. Kyiv: SSU, 1999.

[6]. D. Samoilenko, S. Danylcneko, "Analysis of security aspects of human-oriented names implementations", *Materials of V scientific conference "Modern problems of information security on transport",* pp. 108-109, 2015.

[7]. Server Fault "Faster Option: Redirect via PHP to PHP or Apache mod rewrite redirect to PHP?", 2009. [Online]. Available: https://serverfault.com/questions/ 88919/faster-option-redirect-via-php-to-php-or-apache-mod-rewrite-redirect-to-php [Accessed: May-2017].

ВРЕМЕННАЯ СЛОЖНОСТЬ РАЗЛИЧНЫХ РЕАЛИЗАЦИЙ ДИСПЕТЧЕРА ДОСТУПА СЕТЕВОГО РЕСУРСА

Для создания комплексной системы защиты сетевого информационного ресурса, в соответствии с нормативными требованиями, необходимо реализовывать концепцию диспетчера доступа как единой точки прохождения всех запросов. Существует несколько различных подходов реализации диспетчера доступа, отличающиеся показателями безопасности по надежности. Популярна мысль, что наиболее качественный с точки зрения информационной безопасности вариант является одновременно самым медленными, что ухудшает качество услуги безопасности «доступность». Фундаментальных исследований баланса показателей надежности и доступности по отношению к диспетчерам доступа ранее не проводилось. Реализовано четыре варианта диспетчера доступа средствами сервера Apache и языка программирования PHP; осуществлено экспериментальное определение их временной сложности с помощью аппарата регрессионного анализа. Показано, что различия во временной сложности не превышают 5%, что позволяет отдать предпочтение надежному варианту без ухудшения доступности. Дополнительно засвидетельствовано лучшую временную эффективность языка PHP 7 по сравнению с PHP 5 при реализации клиент-серверного взаимодействия. Использование полученных результатов позволит улучшить показатели качества безопасности сетевых информационных ресурсов.

Ключевые слова: кибербезопасность, сетевой ресурс, диспетчер доступа, Apache, PHP.

TEMPORAL COMPLEXITY OF NETWORK RESOURCE ACCESS MANAGER DIFFERENT IMPLEMENTATIONS

To create a complex data protection system for network information resource in accordance with the regulatory requirements it is necessary to implement the concept of Access Manager as a single transition point of all requests. There are several different approaches to implementation of Access Manager with different reliability as a security parameter. The popular opinion is that the highest security quality Access Manager is also the slowest one. It impairs the other security quality - "availability". Fundamental researches of balance between reliability and availability for Access Managers were not carried out previously. There were implemented four versions of Access Manager for the Apache server and PHP programming language; experimental determination of Access Managers time complexity was performed with linear regression using. It is shown that differences in the time complexity does not exceed 5%, allowing you prefer the most reliable option without compromising availability. Additionally demonstrated better efficiency of language PHP 7 compared to PHP 5 in the implementation of client-server interaction. The use of the results could improve the quality of network security information resources.

Keywords: cybersecurity, network resource, access manager, Apache, PHP.

Самойленко Денис Миколайович, кандидат фізико-математичних наук, доцент, доцент кафедри електрообладнання суден та інформаційної безпеки. Національний університет кораблебудування імені адмірала Макарова.
E-mail: DenNikSam@gmail.com

Самойленко Денис Николаевич, кандидат физико-математических наук, доцент, доцент кафедры электрооборудования суден и информационной безопасности. Национальный университет кораблестроения имени адмирала Макарова.

Samoilenko Denys, PhD, docent of Ship Electrical Equipment and Information Security Department, National University of Shipbuilding after Admiral Makarov.