DOI: https://doi.org/10.15276/aait.03.2020.3
UDC 004.75

REDUCTION OF THE HARMFUL EFFECT OF CRITICAL MODES IN THE OPERATION QUEUE ENVIRONMENT FOR AUTHORIZATION PROTOCOLS FOR LARGE REQUESTS

Sergii S. Surkov
ORCID: http://orcid.org/0000-0001-9224-7526 k1x0r@ukr.net, k1x0r@ukr.net.Scopus ID: 57103247200
Odessa National Polytechnic University, 1, Shevchenko Av. Odessa, 65044, Ukraine

ABSTRACT

An essential part of web security is keeping the payload intact from changes. The data during transmission could be changed, where the encryption is not used, or the data gets decrypted in the middle of the transmission. In our previous researches, the "chunking" method was introduced, which was compared with the "Buffering to file" method. The comparison showed the reduction of recourse consumption. In a multithreading environment, to manage resources efficiently, it is vital to distribute the workload among processor cores. A decent solution for using multithreading efficiently is operation queues. However, if too many operations are accumulated in the operation queue, the system falls into the critical mode. It is characterized by the increase of memory consumption, which may cause the instability of the system. In the course of the study, the main parameters were determined, influencing the data processing speed, and insignificant ones were excluded from the calculation. Earlier, a method was developed for determining the conditions for the falling of a system to a critical mode. It was used as a starting point for the experimental research. A new method based on the method of identifying critical modes in the operation queue is proposed. It differs from existing ones by the ability to simulate critical modes at a given workload, which allows predicting critical modes in order to reduce their negative effect. A series of experiments were carried out, and the results were used to study the dependences of memory consumption on the number of connections and writing speed in critical modes. From the study, three types of critical modes were determined. This made it possible to establish the patterns of the emergence of critical modes in information systems and their impact on the available memory. The formulas are obtained that approximate the experimental data: the dependence of the used memory on the number of connections and the write speed. The research results can be used in the development of information systems and the analysis of failures.

Keywords: Digital Signature; Authorization; Large Payload; Operation Queues; Network Requests; Verification

For citation: **Sergii S. Surkov. Reduction of the Harmful Effect of Critical Modes in the Operation Queue Environment for Authorization Protocols for Large Requests. *Applied Aspects of Information Technology.* 2020; Vol.3 No.3: 145–153. DOI: https://doi.org/10.15276/aait.03.2020.3**

INTRODUCTION

Modern security of web sites uses AI in areas without mandatory registration, where the emphasis is on determining whether a user is a "bot" or a real person. After authenticating the user, the primary security aspect shifts to the authorization of requests from the user to the server. Data transmission can occur through many network nodes that are trusted, but in some nodes, certificates may be substituted, or data is transmitted in an unencrypted form. It makes it necessary to verify the request payload for video streaming, document storage, database services with complex data center infrastructure [1], etc. During the payload verification process, operation queues may experience critical modes leading to the unstable work of the information system.

LITERATURE REVIEW

A crucial part of web security is keeping the payload intact from changes [2]. The generally accepted protocols for communication between clients and servers are HTTP and HTTP/2 [3, 4], which are now widely used not only in web browsers because of their simple and straightforward structure. The generally accepted solution is to use the TLS protocol [5, 6] to verify the encryption and integrity of the data.

However, there are vulnerabilities on the data transmission path, such as ordinary, transparent, or reverse proxies [7, 8], [9], which remove encryption or replace TLS certificates [3], [10, 11]. They are considered to be trusted and managed by data centers or the companies in which the user works.

Attacks such as MITM (Man in the Middle) [12, 13], [14, 15] make it necessary to verify the payload, since not all authorization protocols guarantee data immutability during transmission. The main authorization protocols only check the request headers [16, 17], [18, 19], [20, 21], [22]. Hence the request payload is not authorized in any way. The change of the request payload during transmission can occur if no encryption is used or the data is decrypted during transmission.

The HMAC method is used to prevent modification of the request payload [23]. It is used by authorization protocols such as OAuth 1.0a [24] and HAWK [25, 26], [27]. However, the existing protocols are great for authorizing small payload sizes.

Authorization protocols use payload verification methods for checking the integrity of the payloads of requests. The generally accepted method is “filling the buffer” with its sub-types of “buffering to memory” and “buffering to file” for large and small payloads. We developed the “chunking” method [28, 29], which was recognized as the most promising. The conclusion was made using the method of ranking the implementations of payload signatures [30].

In a multithreaded environment, it is appropriate to distribute the workload across the processor cores to manage the resources efficiently. Operation queues are a suitable solution for this [31]. Operation queues provide efficient operation in a multithreaded environment, allowing large payloads to be authorized.

However, during the overflow of the operation queue, the system goes into critical mode. In the critical mode, memory consumption increases, which may cause system instability? Earlier, a method for identifying critical modes was developed [30], which allows increasing the speed and reliability of the system by identifying the critical workload of the system. The workload on the system can be limited by using the method of migration from a single server to a server cluster.

The means of identifying and preventing critical modes don't get rid of them completely. The study of critical modes makes it possible to predict their impact on equipment and thereby increase the reliability of the system and reduce their harmful effect.

The behavior of the critical modes for authorization protocols for large requests isn't researched well, and their study is an actual task.

THE PURPOSE OF THE ARTICLE

The purpose of this paper is to improve the accuracy of predicting the transition of the system to the critical mode to reduce its harmful effect. Within the article, a method is developed to study the impact of critical modes on the consumption of system memory.

To accomplish the goal, the following tasks were defined:

1) Analyze the critical modes in the operations queue environment for authorization protocols that process large requests.

2) Develop a method for studying the impact of critical modes on the consumption of system memory.

3) Study the effect of the number of connections and writing speed on the falling of the system into the critical mode.

4) Extend the technique and investigate the impact on the system memory.

5) Analyze the dependence of memory consumption on the number of threads and write speed in critical modes.

MAIN PART. OVERVIEW OF CRITICAL MODES IN THE OPERATIONS QUEUE ENVIRONMENT FOR AUTHORIZATION PROTOCOLS THAT PROCESS LARGE REQUESTS

There are several payload verification methods. The first method is “filling the buffer”, the principle of which is to fill the buffer during the downloading the payload [28–29]. The method is shown in *Fig. 1*.

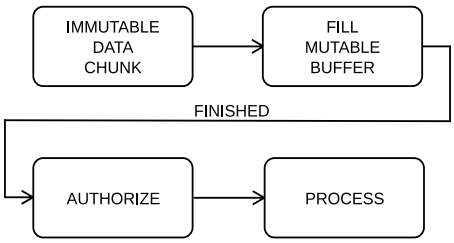

Fig. 1. **Buffer filling method**
***Source*: compiled by the author**

It comes in two variants – “buffering to memory” and “buffering to file”. The first is designed for authorizing a small payload size, the second for a large payload size. It should be noted that for the “buffering in memory” method, the allocation of large blocks of memory is very resource-intensive for the CPU [1], [32, 33], [34], and can lead to unnecessary delay in request processing.

For the large payload size, the “chunking” was established as the most promising. The advantage of the “chunking” method is that it processes chunks that are in the computer's RAM, and that's why the payload is read once. Worth noting that due to file caching by the operating system, the results between “chunking” and “buffering to file” may not differ significantly for small payloads.

The chunking method is shown in Fig. 2.

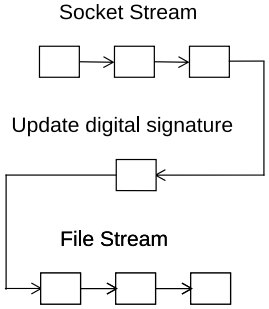

Fig. 2. **“Chunking Method”**
***Source*: compiled by the author**

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)

By serving many requests at the same time, multithreading has an enormous advantage if the HTTP request data is being archived or encrypted in the process. Operation queues are used to process data sequentially while maintaining the benefits of multithreading. With the arrival of each new chunk of data, an operation is created, which is added to the parallel queue and then executed sequentially.

Processing chunks of data (OP) in the operation queue is shown in *Fig. 3.*

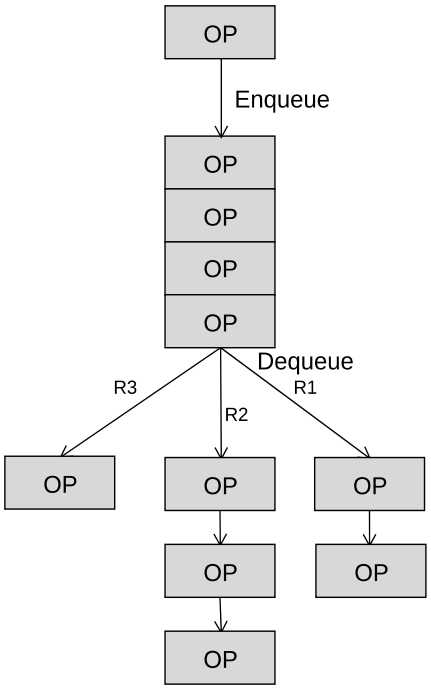

Fig. 3. **Processing chunks in the operation queue**
***Source*: compiled by the author**

HTTP requests R1, R2, R3 are processed sequentially in their operation queues before they are processed in parallel.

However, queued operations may accumulate if the system does not have time to process them, which makes the system fall to the critical mode. Critical mode increases memory consumption, which can lead to system instability.

In the previous paper [31], the method of identifying critical modes was developed. While developing that method, it was intended to reuse its elements to study the critical modes themselves.

THE METHOD FOR THE STUDY OF THE IMPACT OF CRITICAL MODES ON THE SYSTEM'S MEMORY

The basis of our queueing system is libdispatch [35], which provides comprehensive support for concurrent code execution on multicore hardware. The framework is available for Apple and Linux platforms.

A new method for studying the impact of critical modes on the system's memory in the operation queue environment is proposed, based on the method of identifying critical modes [30] and simulation of asynchronous operations.

This method differs from the existing ones by the ability to simulate critical modes at a given workload, which makes it possible to increase the accuracy of predicting the transition of the system to the critical mode to reduce their harmful effect.

The difference is provided by the fact that the new method uses operation queues directly to process input data. With a specified workload, this allows us to investigate the tendency of critical modes.

The method includes the following steps:

1) initiate the launch of a predetermined number of connections at the same time;

2) add to the operation queue chunks of data with a given rate;

3) measure the rate of data processing in the operation queue environment;

4) measure the amount of memory occupied by the server process every selected time interval;

5) display the results of measurement.

For methods of ranking payload verification implementations and identifying critical modes, BenchmarkChunking and BenchmarkFileBuffering classes were created. They are inherited from BenchmarkBase class [31]. For the new method of studying the impact of critical modes on the payload verification implementations in the operation queue environment, a new class BenchmarkCriticalModes is created.

The benchmark method of each instance of the BenchmarkCriticalModes object runs on a separate thread. During the initialization of each instance of the class, the following variables passed through the constructor during initialization (*Fig. 4*)

```
FILE* file;
std::string filePath;
dispatch_queue_t queue;
```

Fig. 4. **Variables initialized in the constructor of the BenchmarkCriticalModes class**
***Source*: compiled by the author**

For the asynchronous test, all the chunks are submitted to the queue sequentially with the delay of ChunkTransmissionTimeMs, which simulates network transfer speed. After all the chunks are processed, the statistics are displayed.

In the new method, measuring the speed and time of the processing of a chunk is similar to the method for identifying critical modes. In the method, to achieve many connections, thread per connection pattern is used to emulate concurrent connections.

The implementation of the queuing system for the technique in C++ is in *Fig. 5*.

```
virtual void benchmarkQueue() {
  dispatch_semaphore_t s =
            dispatch_semaphore_create(0);
  for (ssize_t i = 0; i < NChunks; i++) {
      __block uint8_t* data =
            (uint8_t *) malloc(ChunkSize);
      dispatch_async(queue, ^{
          auto writeMs = benchmarkChunk(data);
          updateTotalTime<false>(writeMs);
          free(data);
      });
      sleep_us(timeDiff);
  }
  dispatch_async(queue, ^{
      dispatch_semaphore_signal(s);
  });
  dispatch_semaphore_wait(s, DISPATCH_TIME_FOREVER);
  fclose(file);
  printStatsForThread();
  remove(filePath.c_str());
}
```

Fig. 5. **Function of measuring the impact of the critical modes on the system's memory**

***Source*: compiled by the author**

STUDYING THE IMPACT OF THE NUMBER OF CONNECTIONS AND WRITING SPEED ON THE TRANSITION OF THE SYSTEM TO CRITICAL MODE

Since our newly developed method uses operation queues, the system may fall into critical mode. It may happen because the stack of operations is growing over time.

In case if the server process will use more memory than possible, then the operating system will shut down the process. It is sufficient to reduce the amount of RAM to the top point of the graphs from the experiments to have the server process terminated. The following equipment was used for the tests:

OS: Ubuntu 18.04 LTS

CPU: Core i7 8700K

RAM: 32G

SSD: Samsung 960 Evo 512G

Two series of experiments were carried out to study the critical modes. The first series of experiments assume a client speed of 100 Mbps. According to the method of identifying critical modes, from 32 to 48 connections were selected.

Each line in the graph is an experiment showing memory consumption with a given number of connections. The more connections, the more memory the system needs to process data in critical mode. Observing the plot, the critical mode each line has 3 types: in the beginning there is a linear growth. This happens because drive can't keep up with incoming data.

The second type is when server's drive is able to keep up with incoming data.

The third type is when no data is coming and the entire buffer is being flushed to the drive.

Due to libdispatch uses ARC (Automatic Reference Counting) mechanism, the decrease of used memory is discrete.

The results for the series of experiments for input bandwidth of 100 Mbps per client are in *Fig. 6*.

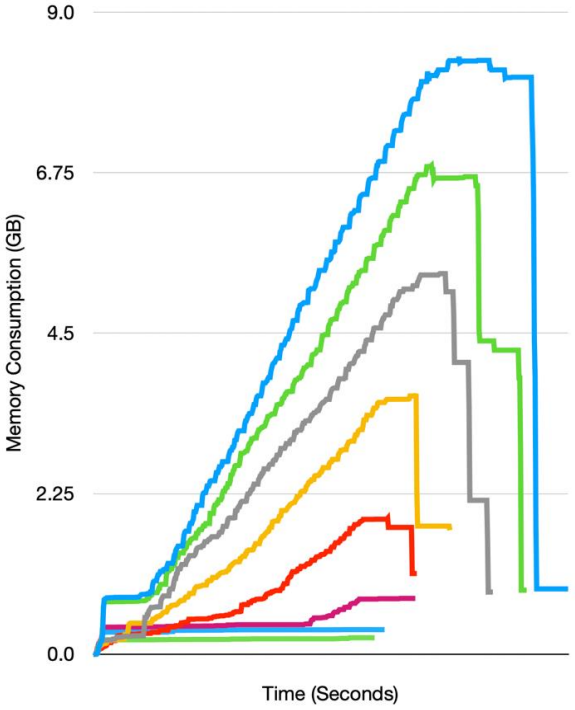

Fig. 6. **Critical mode for the input bandwidth of 100 Mbps per client**

***Source*: compiled by the author**

For the second series of experiments, a connection speed of 1 Gbps was chosen. According to the method of identifying critical modes, from 5 to 10 connections were selected.

The results for the series of experiments for 1 Gbps are in Fig. 7.

Having conducted the series of experiments, the data needs to be processed to find the number of threads, from which the system goes into the critical mode.

ANALYSIS OF THE DEPENDENCY OF MEMORY CONSUMPTION ON THE NUMBER OF THREADS AND WRITE SPEED IN CRITICAL MODES

To summarize the data in *Fig. 8* and *Fig. 9*, summary graphs were built showing the dependence of the maximum memory consumption on the number of threads. The dots on the graph show the experimental data, the calculation by the formula is shown by the solid line.

The experimental and theoretical results for the 100 Mbps series of experiments is in Fig. 8.

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)

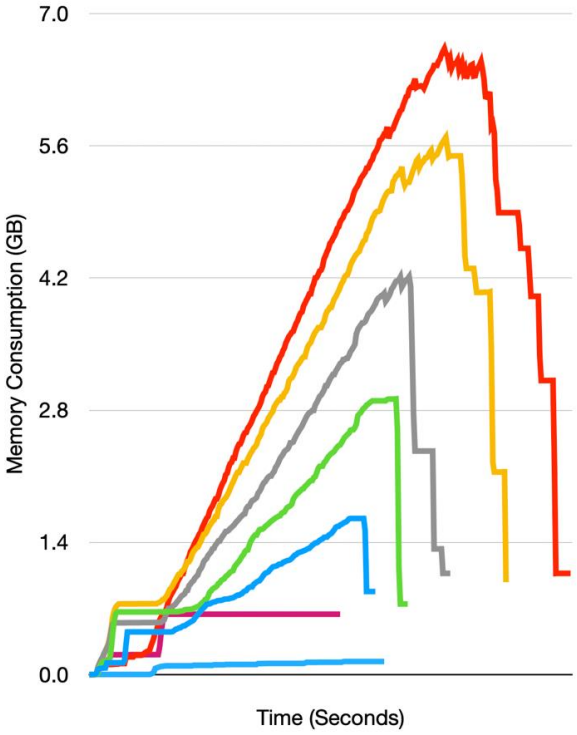

Fig. 7. **Critical mode for the input bandwidth of 1 Gbps per client**
***Source*: compiled by the author**

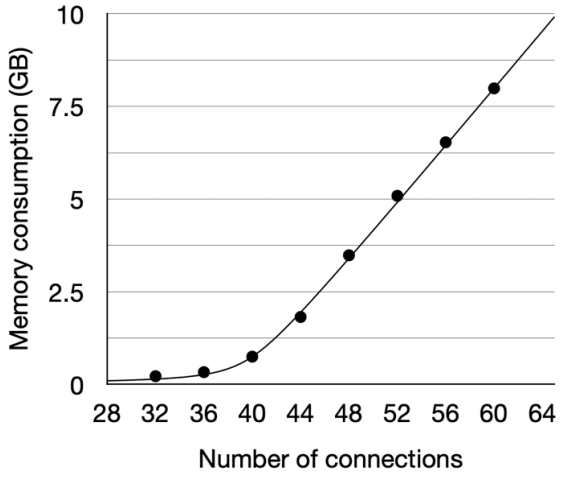

Fig. 8. **Maximum memory consumption for the series of experiments for 100 Mbps (12.5 MB/s)**
***Source*: compiled by the author**

The experimental and theoretical results for 1 Gbps series of experiments is in Fig. 9.

As can be seen from Fig. 8 and Fig. 9, the dependence of the maximum memory consumption on the number of threads can be described by the hyperbola equation:

$$x = -\frac{c}{y} + a \cdot y + b\,.$$

For the convenience of calculations, it is advisable to obtain an inverse relationship y(x).

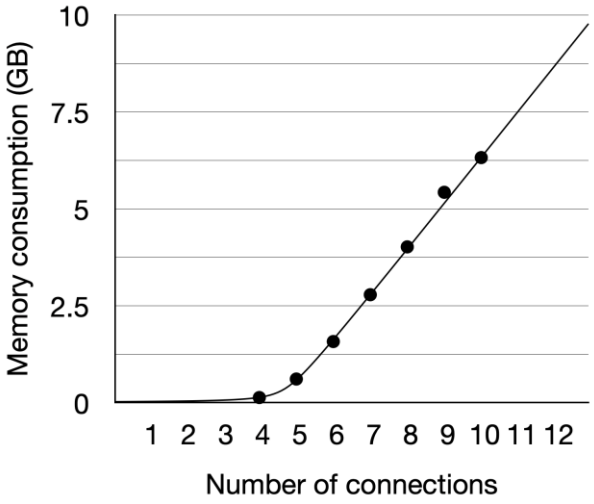

Fig. 9. **Maximum memory consumption for the series of experiments for 1 Gbps (125 MB/s)**
***Source*: compiled by the author**

Solving the quadratic equation gives the following formula:

$$y = \frac{-(b-x) + \sqrt{(b-x)^2 + 2 \cdot a \cdot c}}{a}\,.$$

The coefficients in this equation are calculated by multivariate optimization to achieve the least sum of squared deviations.

These coefficients received the following physical interpretation:

a – how fast the system goes into critical mode

b – number of threads, from which system goes into critical mode;

c – the growth of memory before the system falls into critical mode;

x – number of threads;

y – maximum memory consumption.

The unbiased estimate of the standard deviation (σ) is calculated using the formula:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} e_i^2}{n-1}}\,.$$

For the best match with the experimental points, for each series of experiments, the following coefficients were determined:

(100 Mbps or 12,5 MB/s)
a = 5,148, b = 39,58, c = 1,12,
σ = 0,1002 GB
(1 Gbps or 125 MB/s),
a = 1,68 b = 4,7 c = 0,112,
σ = 0,1199 GB.

The analysis of the coefficients shows that the critical mode starts with 39 active connections (for

100 Mbps per client) and with 4 active connections (for 1 Gbps per client). Such values of the unbiased estimate of the standard deviation (σ) indicate the acceptable accuracy of the approximation.

The connection speed (S) is multiplied by the number of connections at which the system goes to critical mode (b) to find the maximum concurrent write speed (P_W).

To calculate the required parallel write speed (R_W), the connection speed (S) multiplies by the number of connections (C) from the experiments:

$$P_W = b * S,$$
$$R_W = C * S.$$

The results the series of experiments of 100 Mbps (12.5 MB/s):

$P_W = 12,5 * 39,58 = 494,75$ MB/s,
$R_W = 12.5 * 60 = 750$ MB/s.

The results for the series of experiments of 1 Gbps (125 MB/s):

$P_W = 125 * 4,7 = 587,5$ MB/s,
$R_W = 125 * 10 = 1250$ MB/s.

The deviation of the maximum parallel write speeds between the series of experiments is 100 MB/s. With very different required parallel write speeds, it is concluded that the maximum parallel write speed is weakly dependent on the number of connections in the operation queue environment.

To confirm this assumption, we need to investigate the dependence of the maximum used memory in the critical mode on the overall speed. Based on the results above, a series of experiments were carried out, in which the incoming speed varies from 400 to 640 MB/s. The number of compounds in all experiments was 16.

The results of the experiment are in *Fig. 10*.

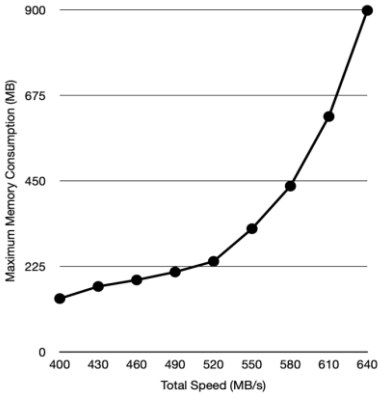

Fig. 10. **Maximum memory consumption for the series of experiments for 100 Mbps (12.5 MB/s)**
***Source*: compiled by the author**

The experiment showed that at the total write speed of 520 MB/s the server process goes into critical mode. From this point, a sharp increase in memory consumption begins. With the connection speed per client from 25 to 37.5 MB/s (200 to 300 Mbit/s) with the same number of threads, the experiment confirms that in the operation queue environment, the total speed plays a more significant role than the number of connections.

From the conducted research, the following statements are made:

1) If the system does not have time to process operations in the queue, the system may go into critical mode.

2) There may be a regime of "stable consumption" when the incoming speed is approximately the same as writing speed.

3) If the system managed to process the accumulated data, the server process returns into the normal mode.

4) Limiting upload speed across all threads generally eliminates falling of the system into the critical mode.

5) The research carried out in this article can be useful in designing an information system and adjusting its parameters in order to avoid the system falling into a critical mode.

CONCLUSIONS

A new method was developed to study the influence of critical modes on the payload verification implementations in the operation queue environment. A series of experiments were carried out using this method. The results of them were used to investigate the dependence of the used memory on the number of connections and the write speed in critical modes.

From the study, three types were determined. It's become possible to establish the patterns of the emergence of critical modes in information systems and their impact on the available memory. The formulas are obtained that approximate the experimental data on the dependence of the used memory on the number of connections and write speed. The research results can be used in the development of information systems and the analysis of failures.

The new method and the conducted research provide an increase in the accuracy of predicting the transition of the system to the critical mode. Thus, the set goal of the study has been achieved.

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)

REFERENCES

1. Surkov, S. & Martynyuk, O. “Method of Migration from Single Server System to Server Cluster”. In *Proceedings of the 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS’2015)*. Warsaw: Poland. 2015. DOI: https://doi.org/10.1109/IDAACS.2015.7341415.
2. Kizza, J. M. “Computer Network Security and Cyber Ethics Fourth Edition”. Jefferson, NC. United States: *Publ. McFarland.* 2014. 240 p.
3. Fielding, R. & Reschke, J. “ Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing, IETF RFC 7230”. – Available from: https://tools.ietf.org/html/rfc7230.– [Accessed 12th August 2020].
4. Belshe, M. & Peon, R. “Hypertext Transfer Protocol Version 2 (HTTP/2), IETF RFC 7540”. – Available from: https://tools.ietf.org/html/rfc7540. – [Accessed 12th August 2020].
5. Liu, Q., Zhang, L. & Fan, A. “Scheme to Authenticate Requests for Online Banking Based on Identity-based Mediated RSA”. *Jiefangjun Ligong Daxue Xuebao/Journal of PLA University of Science and Technology (Natural Science Edition).* 2015; Vol. 16: 29–33. DOI: https://doi.org/10.7666/j.issn.1009-3443.20140929001.
6. Sanae, H. “Security Requirements and Model for Mobile Agent Authentication”. *Smart Network Inspired Paradigm and Approaches in IoT Applications* Singapore: *Republic of Singapore.* 2019. p. 179–189. DOI: https://doi.org/10.1007/978-981-13-8614-5_11.
7. Rash, M. “Linux Firewalls: Attack Detection and Response with iptables, psad, and fwsnort”. San Francisco, CA, United States: *Publ. No Starch Press.* 2007. 336 p.
8. Fjordvald M. & Nedelcu C. “Nginx HTTP Server – Fourth Edition: Harness the Power of Nginx to Make the Most of Your Infrastructure and Serve Pages Faster Than Ever Before”. Birmingham, United Kingdom: *Publ. Packt Publishing.* 2018. 400 p.
9. Blokdyk, G. “Apache Web Server A Complete Guide – Edition”. Brisbane, Australia: *Publ. 5STARCooks.* 2020. 238 p.
10. Saini, K. “Squid Proxy Server 3.1: Beginner's Guide Paperback”. Birmingham, United Kingdom: *Publ. Packt Publishing.* 2011. 332 p.
11. Wessels, D. “Squid: The Definitive Guide”, Sebastopol, CA, United States: *Publ. O'Reilly Media.* 2010. 472 p.
12. Eugene, F., John, O.R. & Kevin, C. “Security Evaluation of the OAuth 2.0 Framework”. *Information and Computer Security*. 2015; Vol.23(1): 73–101. DOI: https://doi.org/10.1108/ICS-12-2013-0089.
13. Cheol-Joo, Chae Ki-Bong & Han-Jin Cho. “A Study on Secure User Authentication and Authorization in OAuth Protocol”. *Springer Cluster Computing.* 2019; Vol. 22(2). DOI: https://doi.org/10.1007/s10586-017-1119-6.
14. Farooqi, S., Zaffar, F., Leontiadis, N., et al. “Measuring and Mitigating OAuth Access Token Abuse by Collusion Networks”. In *Communications of the ACM*. New York: NY, United States. 2020. p. 103–111, DOI: https://doi.org/10.1145/3387720.
15. Feng, Y. & Sathiamoorthy, M. “A Security Analysis of the OAuth Protocol”. In *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*. Victoria, BC, Canada: 2013. p. 271–276. DOI: https://doi.org/10.1109/PACRIM.2013.6625487.
16. Seung, J. & Souhwan, J. “Personal OAuth Authorization Server and Push OAuth for Internet of Things”. *International Journal of Distributed Sensor Networks*. Thousand Oaks:, CA, United States. 2017. Vol.13. DOI: https://doi.org/10.1177/1550147717712627.
17. Se-Ra, O. & Young-Gab, K. “AFaaS: Authorization Framework as a Service for Internet of Things based on Interoperable OAuth”. *International Journal of Distributed Sensor Networks*. Thousand Oaks: CA, United States. 2020; Vol.16(2): 1–15. DOI: https://doi.org/10.1177/1550147720906388.
18. Hossain, N., Hossain, M. A., Hossain, M., et al. “OAuth-SSO: A Framework to Secure the OAuth-based SSO Service for Packaged Web Applications”. In *Proc. of 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. New York: NY, United States. 2018. p. 1575–1578. DOI: https://doi.org/10.1109/TrustCom/BigDataSE.2018.00227.
19. El-hajj, M., Fadlallah, A., Maroun, C., et al. “A Survey of Internet of Things (IoT) Authentication Schemes”. *Sensors – Open Access Journal.* Basel: Switzerland. 2019; Vol. 19: 1–17. DOI: https://doi.org/10.3390/s19051141.

20. Hardt, D. “The OAuth 2.0 Authorization Framework, IETF RFC 6749”. – Available from: https://tools.ietf.org/html/rfc6749. – [Accessed 26th Jule 2020].

21. Jones, M. & Bradley, J. “JSON Web Token (JWT) IETF RFC 7519”. – Available from: https://tools.ietf.org/html/rfc7519. – [Accessed 18th Jule 2020].

22. Richer, J. “User Authentication with OAuth 2.0”. – Available from: https://oauth.net/articles /authentication/. – [Accessed 17th Jule 2020].

23. Krawczyk, H. & Bellare, M. “HMAC: Keyed-Hashing for Message Authentication”. – Available from: https://tools.ietf.org/html/rfc2104. – [Accessed 02th Jule 2020].

24. Leiba, B. “OAuth Web Authorization Protocol”. *IEEE Internet Computing*. Nicosia: Cyprus. 2012; Vol.16: 74–77. DOI: https://doi.org/10.1109/MIC.2012.11.

25. Hammer-Lahav E.. “The OAuth 1.0 Protocol. IETF RFC 5849”. – Available from: http://tools.ietf.org/html/rfc5849. – [Accesse15th August 2020].

26. Hammer, E. “OAuth 2.0 and the Road to Hell”. – Available from: http://hueniverse.com/2012/07/ oauth-2-0-and-the-road-to-hell/. – [Accessed 29th June 2020].

27. Hammer, E. “HAWK / HTTP Holder-Of-Key Authentication Scheme”. – Available from: https://github.com/hueniverse/hawk. – [Accessed 14th August 2020].

28. Surkov, S. S. “Model and Method of Chunk Processing of Payload for HTTP Authorization Protocols”. *Proceedings of IEEE 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)*, Slavske: Ukraine. 2020. p. 317–321. DOI: 10.1109/TCSET49122.2020.235447.

29. Surkov, S. S., Martynyuk, O. M. & Mileiko, I. G. “Modification of Open Authorization Protocol for Verification of Request“(in Russian). *Electrotechnic and Computer systems*. Odessa: Ukraine. 2015; Vol. 19(95): 178–181.

30. Surkov, S. S. “Comparison of Authorization Protocols for Large Requests in Operation Queue Environment”. *Applied Aspects of Information Technology*. *Publ. Nauka I Tekhnika*. Odesa: Ukraine. 2020; Vol. 3 No.3. DOI: https://doi.org/10.15276/hait.03.2020.5**.**

31. Grosch, S. “Concurrency by Tutorials (Second Edition): Multithreading in Swift with GCD and Operations”, McGaheysville, VA, United States. *Publ. Razeware LLC.* 2020. 100 p.

32. Drozd, O., Kharchenko, V., Rucinski, A., et al. “Development of Models in Resilient Computing”. In *Proc. of 10th IEEE International Conference on Dependable Systems, Services and Technologies (DESSERT’2019)*. Leeds: UK. 2019. DOI: https://doi.org/10.1109/DESSERT.2019.8770035.

33. Drozd, A., Antoshchuk, S., Drozd, J., et al. “Checkable FPGA Design: Energy Consumption, Throughput and Trustworthiness”. In *Green IT Engineering: Social, Business and Industrial Applications, Studies in Systems, Decision and Control*. Warsaw: Poland. 2018. p. 73–94. DOI: https://doi.org/10.1007/978-3-030-00253-4_4.

34. Drozd, O., Kuznietsov, M., Martynyuk, O., et al. “A Method of the Hidden Faults Elimination in FPGA Projects for the Critical Applications”. In *Proc. of 9th IEEE International Conference on Dependable Systems. Services and Technologies (DESSERT’2018)*. Kyiv: Ukraine. 2018. p. 231–234. DOI: https://doi.org/10.1109/DESSERT.2018.8409131.

35. Apple Inc. “Grand Central Dispatch”. – Available from: https://github.com/ apple/swift-corelibs-libdispatch. – [Accessed 27th June 2020].

Conflicts of Interest: the authors declare no conflict of interest.

Received 08.08.2020
Received after revision 14.09.2020
Accepted 21.09.2020

DOI: https://doi.org/10.15276/aait.03.2020.3
УДК 004.75

ЗНИЖЕННЯ ШКІДЛИВОГО ВПЛИВУ КРИТИЧНИХ РЕЖИМІВ У СЕРЕДОВИЩІ ЧЕРГ ОПЕРАЦІЙ ДЛЯ ПРОТОКОЛІВ АВТОРИЗАЦІЇ ВЕЛИКИХ ЗАПИТІВ

Сергій Сергійович Сурков
ORCID: http://orcid.org/0000-0001-9224-7526, k1x0r@ukr.net
Одеський національний політехнічний університет пр. Шевченка, 1. Одеса, 65044,Україна

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)

АНОТАЦІЯ

Важливою частиною веб-безпеки є збереження цілісності корисного навантаження. Зміна даних під час передачі може виникнути, якщо не використовується шифрування, або дані розшифровуються в процесі передачі. У наших попередніх дослідженнях було представлено «порційний» метод, який порівняли з методом «буферізация в файл» і довели, що він зменшує споживання ресурсів. У багатопотоковому середовищі для ефективного управління ресурсами життєво важливо розподіляти робоче навантаження між ядрами процесора. Хорошим рішенням для використання переваг багатопоточності є черги операцій. Однак при переповненні черги операцій система переходить в критичний режим. Для нього характерне збільшення споживання пам'яті, що може привести до нестабільності системи. В ході дослідження були визначені основні параметри, що впливають на швидкість обробки даних, а незначні були виключені з розрахунку. Раніше був розроблений метод визначення умов переходу системи в критичний режим, який був використаний в якості відправної точки для експериментальних досліджень. Запропоновано новий метод дослідження, заснований на методі визначення переходу системи в критичний режим і імітаційному моделюванні асинхронних операцій. Даний метод відрізняється від існуючих роботою з чергами операцій безпосередньо, що дозволяє прогнозувати критичні режими з метою зменшити їх негативний ефект. Наступна серія експериментів, дозволила вивчити залежності споживання пам'яті від кількості з'єднань і швидкості запису в критичних режимах. За результатами дослідження встановлено, що існує три основних типи. Це дозволило встановити закономірності виникнення критичних режимів в інформаційних системах і їх вплив на доступну пам'ять. Отримано формули, апроксимуючі експериментальні дані про залежність обсягу споживаної пам'яті від кількості з'єднань і швидкості запису. Результати дослідження можуть бути використані при розробці інформаційних систем і при аналізі збоїв в їх роботі.

Ключові слова: цифровий підпис; авторизація; великий розмір запиту; черги операцій; мережеві запити; верифікація;

DOI: https://doi.org/10.15276/aait.03.2020.3
УДК 004.75

СНИЖЕНИЕ ВРЕДНОГО ВОЗДЕЙСТВИЯ КРИТИЧЕСКИХ РЕЖИМОВ В СРЕДЕ ОЧЕРЕДЕЙ ОПЕРАЦИЙ ДЛЯ ПРОТОКОЛОВ АВТОРИЗАЦИИ БОЛЬШИХ ЗАПРОСОВ

Сергей Сергеевич Сурков
ORCID: http://orcid.org/0000-0001-9224-7526, k1x0r@ukr.net
Одесский национальный политехнический университет, пр. Шевченко, 1. Одесса, 65044, Украина

АННОТАЦИЯ

Важной частью веб-безопасности является сохранение целостности полезной нагрузки. Изменение данных во время передачи может возникнуть, если не используется шифрование, или данные расшифровываются в процессе передачи. В наших предыдущих исследованиях был представлен «порционный» метод, который сравнили с методом «буферизация в файл» и доказали, что он сокращает потребление ресурсов. В многопоточной среде для эффективного управления ресурсами жизненно важно распределять рабочую нагрузку между ядрами процессора. Хорошим решением для использования преимуществ многопоточности являются очереди операций. Однако при переполнении очереди операций система переходит в критический режим. Для него характерно увеличение потребления памяти, что может привести к нестабильности системы. В ходе исследования были определены основные параметры, влияющие на скорость обработки данных, а незначительные были исключены из расчета. Ранее был разработан метод определения условий перехода системы в критический режим, который был использован в качестве отправной точки для экспериментальных исследований. Предложен новый метод исследования, основанный на методе определения перехода системы в критический режим и имитационном моделировании асинхронных операций. Данный метод отличается от существующих работой с очередями операций непосредственно, что позволяет прогнозировать критические режимы с целью уменьшить их негативный эффект. Следующая серия экспериментов, позволила изучить зависимости потребления памяти от количества соединений и скорости записи в критических режимах. По результатам исследования установлено, что существует три основных типа. Это позволило установить закономерности возникновения критических режимов в информационных системах и их влияние на доступную память. Получены формулы, аппроксимирующие экспериментальные данные о зависимости объема потребляемой памяти от количества соединений и скорости записи. Результаты исследования могут быть использованы при разработке информационных систем и при анализе сбоев в их работе.

Ключевые слова: цифровая подпись; авторизация; большой размер запрпоса; очереди операций; сетевые запросы; верификация

ABOUT THE AUTHOR

Sergii S. Surkov – PhD Student of Computer Intellectual Systems and Networks Department, Odessa National Polytechnic University, 1, Shevchenko Av. Odessa, 65044, Ukraine
k1x0r@ukr.net, ORCID: http://orcid.org/0000-0001-9224-7526

Сергій Сергійович Сурков – аспірант каф. комп'ютерних інтелектуальних систем і мереж. Одеський національний політехнічний університет пр. Шевченка, 1. Одеса, 65044,Україна