Article

Resource Analysis of the Log Files Storage Based on Simulation Models in a Virtual Environment

Shamil Magomedov, Dmitry Ilin and Evgeny Nikulchev *

Department of Intelligent Information Security Systems, MIREA—Russian Technological University, 119454 Moscow, Russia; magomedov_sh@mirea.ru (S.M.); i@dmitryilin.com (D.I.)

* Correspondence: nikulchev@mail.ru

Abstract: In order to perform resource analyses, we here offer an experimental stand on virtual machines. The concept of how to measure the resources of each component is proposed. In the case of system design, you can estimate how many resources to reserve, and if external modules are installed in an existing system, you can assess whether there are enough resources and whether the system can scale. This is especially important for large software systems with web services. The dataset contains a set of experimental data and the configuration of virtual servers of the experiment in order to conduct resource analyses of the logs.

Keywords: SIEM systems; component resource analysis; experimental stand on virtual machines

1. Introduction

SIEM systems are a developing area in the field of computer security. However, incorporating a new storage component into computing infrastructure is often difficult. In this situation, a system administrator needs to know how many resources are required for a particular SIEM component.

A significant amount of modern research has been devoted to the development of SIEM system architectures [1,2], the identification of threat sources and mechanisms for their detection in distributed systems [3–5], to block malicious traffic from IoT devices [6], and to research intellectual data processing from several sources [7,8], using event classification methods [9,10]. Other research directions are dedicated to the extraction of data that are applicable for access control [5,11,12], in the analysis of user behavior [13], and for computer security specific situations in different environments (e.g., IoT, smart cities) [11,14,15].

In modern research, considerable attention is paid to the architecture of computing systems when introducing access control tools. The consideration of security issues in the organization of access control, separately from the computing complex for high-load multi-user services (diagram shown in Figure 1) entails the problems of choosing and configuring network and server equipment and the parameters of data storage systems [16,17]. Additional security measures might have an impact on computing system resource efficiency, so it is reasonable to evaluate the impact prior to production deployment of a particular SIEM component. Thus, the aim of the study is to select tools and models that provide a solution to the problems of building a computing system architecture for web services that implement resource-efficient technological solutions.
Specific features of the use of web portals, web services, and mobile applications is the impossibility of determining the intensity of user requests of the complete infrastructure itself (since it is not possible to determine all stages of signal transmission through networks since this includes providers’ servers caching data processing servers and other network equipment not connected in any way to the designed computing complex). In addition, existing commercial SIEM solutions analyze access at the level of the network [18] and data center servers, which excludes a number of important points that require control, including those associated with user behavior on client devices, and these data can only be obtained at the level of developed software. All this requires the development of a specialized architecture, methods, and models for the research object under consideration.

Resource efficiency can be analyzed with approaches such as algorithm complexity analysis [19] and benchmarking [20]. The former can provide information on the execution time of an algorithm, yet it is hardly applicable for the analysis of complex software systems. The latter is a well-known approach for software resource efficiency analysis, yet it does not consider the specifics of computing infrastructure nor the planned volume of data requests. To solve the problem of building an efficient computing service architecture, considering the given specifics, it is necessary to develop a methodology for simulation modeling to assess the values of the component parameters of the SIEM system.

2. Materials and Methods

The task of building an architecture is to build a set of components that implement SIEM and their interconnections, so that there is an opportunity to assess their resource efficiencies. In other words, for the flow of user requests, \( x \), to a web service and for the component, \( Z_i \), of the architecture, \( V \), the following can be estimated:

\[
Z_i \subset V : x \xrightarrow{\Phi} R_i, \quad i = 1, q,
\]  

(1)

where \( R_i = \mathbb{R}^n \) is the vector of the measured computing resources; \( n \) is the dimensionality of this vector; \( \Phi \) is a mapping such that parameter \( R_i \) is measurable for the observed process (x). That is, the architecture must ensure that the \( \Phi \) mapping is identifiable for a given stream of events. This can be achieved by implementing the dependency of each component on the input stream and the ability to measure resources.

To estimate the resources \( (R_i) \) in expression (1), due to the structural transparency of the architecture, simulation modeling can be used. In a number of works devoted to the construction of the mathematical models of web-portal processes, it is shown that stochastic processes describing user access can be identified based on typical requests. However, the
use of the dynamic stochastic models used in these works does not seem appropriate for solving the problems of assessing the resource efficiency of the components of the access control architecture. In general, dynamic models are widely used to build traffic models. To estimate resources, there is no need to build accurate predictive models of processes, since the sought-after values of the stocks of computing resources depend only on the range and intensity of processes, which can be implemented via simulation of a random variable with a given distribution function. Note that, when using public networks with the TCP/IP protocol, due to the limited channel, an increase in the frequency of requests is observed in the histogram in the area of the upper boundary, which is associated with re-sent non-missed packets, that is, heavy-tailed distributions are observed.

A method for analyzing the costs of computing resources for the implementation of access control systems has been developed, based on an approach that uses virtual stands that provide a simulation environment that uses a computing complex at each level of access control. The technique consists of seven steps:

1. The building of a typical user request.
2. Implementation of an access control system with the means for data flow control, generated by a typical request.
3. Creation of a virtual experimental stand that simulates an environment for using architecture components.
4. Formation of a random signal with a given distribution law based on typical user requests.
5. Obtaining estimates of the values of the resources required to use the access control system.
6. In the case of solving the problem of choosing options for the implementation of CA means, the selection of options that have lower resource costs.
7. Formation of the architecture of the computing complex, taking the obtained values of the costs of computing resources into account.

Consider, for example, the problem of evaluating computational costs when recording events in a database while working with web services over computer networks with logging user actions. It is assumed that the recording of user actions will be carried out in the event log. For each device, keeping a record of the activity log is proposed. For experimental research, we will use the following data: the volume of the original data file is 460 MB without formatting and the file contains records in a semi-structured JSON format.

Before starting the experiment, three virtual machines (VM) (client, server, and database) with specified characteristics were created. VirtualBox was utilized as a hypervisor for the task. RAM and CPU allocation policies were left at default settings. For repeated experiments, previously created VMs were deleted and then created anew to ensure the exact same experimental conditions between repetitions. VM management was conducted using Vagrant, while VM provisioning was implemented with Ansible. The host machine hardware was as follows:

- **CPU**—AMD Ryzen 7 3700X 8-Core Processor, 3600 MHz, 8 physical cores, 16 logical cores.
- **RAM**—32 Gb DDR4, frequency 1600 MHz, Dual Channel Mode.
- **Disk Subsystem**—Samsung SSD 970 EVO Plus.

All VMs were ran the same guest operating system: Ubuntu 16.04 LTS (Xenial Xerus), Vagrant box 20190816.0.0. The structure of the experimental stand is shown in Figure 2 and in Table 1.
Table 1. Parameters of the virtual machines.

| Resource       | CPU Cores | RAM (MB) | Maximum Allowed Load of CPU Cores (%) | Input–Output System Bandwidth (MB/sec) |
|----------------|-----------|----------|--------------------------------------|---------------------------------------|
| Client         | 4         | 8192     | 100                                  | –                                     |
| Server         | 2         | 2048     | 100                                  | –                                     |
| Database       | 2         | 2048     | 50                                   | 25                                    |

3. Results

After creating a VM, and installing and running the server software and DBMS, the experiment began. The initial data were loaded into the RAM of the client machine. After they were fully loaded, the sending of data with the specified parameters began.

The experiment is discussed in detail in Appendix A.

The results of the computational experiment are shown in Table 2.

Table 2. Indicators of resource costs for logging user actions.

| Resource Indicator     | Value without Using Logging | Value with Using Logging | Difference in % |
|------------------------|-----------------------------|--------------------------|-----------------|
| Client VM CPU          | 8.498                       | 8.217                    | 3.3             |
| Server VM CPU          | 19.710                      | 22.230                   | 12.7            |
| Database VM CPU        | 2.859                       | 4.371                    | 52.9            |
| Client VM Free RAM     | 1296,828.186                | 1295,666.406             | 0.08            |
| Server VM Free RAM     | 41,345.797                  | 39,147.056               | 5.32            |
| Database VM Free RAM   | 340,911.115                 | 341,359.788              | 0.13            |

Thus, for the considered example, the use of logging user actions was experimentally established and only significantly affected the load on the server processor (Figure 3) and the database processor (Figure 4), and insignificantly increased the server memory load. The introduction of access control with logging requires the introduction of appropriate resources reserves into the computing complex.
4. Data Description

The dataset (http://dx.doi.org/10.17632/25v6shzfff.1, accessed on 27 February 2021) consists of three files:

1. File with input data (initial-dataset.json), which is used by the client to send requests.
2. File with the results of monitoring virtual machine resources for the experiment without logging (monitoring-data_wo-logging.json).
3. File with the results of monitoring virtual machine resources for the experiment with logging (monitoring-data_w-logging.json).

The input data file contains two types of documents: ResearchSubject and ResearchResult. They are related in a one-to-many relationship, i.e., a ResearchSubject can have zero or more associated ResearchResult documents. In the source file, this link is implemented through nesting.

The ResearchSubject document contains 8 string type attributes and a single attribute representing the array of ResearchResult documents. ResearchResult document consists of 7 string type attributes, a single numeric type attribute and an attribute named “data”. The “data” attribute content represents an object of various structures. The document examples are available in Appendix B and Figures A3 and A4, respectively. As the part of the experiment with the addition of logging for each incoming request, the server software generates ActionLog documents, the structure of which is shown in Appendix B, Figure A5. These documents contain basic information on HTTP requests, such as user identifiers, Boolean flag marking a user’s existence in the system, time of the log entry creation, the HTTP method, and the called remote method name. They are generated programmatically.

Figure 3. Used CPU on server VM, %.

Figure 4. Used CPU on database VM, %.
and a hook is used (the source code is shown in Appendix B, Figure A6), which is triggered after the received request is completed. It is important to note that all requests in the experiment contain a userId, so, after each request, two consecutive actions are performed:

- the user with the userId identifier is found;
- an ActionLog document is created and written to the database.

The output from monitoring virtual machine resources is provided by the atop utility. Raw output of the atop utility requires the making of a domain-specific parser, as it does not conform to widespread formats, such as CSV, etc. In addition, the documentation refers to specific data columns by their position, which is hard to read. To provide data in a more usable form, the raw output was converted into the JSON format.

The output files have an identical structure (see Appendix B, Figure A7 for a generalized example). Each file contains three objects with data available under the following keys: “client” is for the results of monitoring the resources of the client VM, “server” is for the results of monitoring the resources of the server VM, and “mongodb” is for the results of monitoring the resources of the DBMS VM.

5. Conclusions

This paper considered the task of the selection of resource-efficient technological solutions for building a computing system architecture for web services. The SIEM systems application scope was investigated as an important area in the field of computer security.

A method for analyzing the costs of computing resources for the implementation of access control systems has been proposed. An experiment was conducted using the proposed method. The experimental details are given, as well as the corresponding datasets and experiment virtual environment configurations. It was demonstrated that the SIEM component resource efficiency impact can be measured using virtual environments.

Development of frameworks and automation tools for resource efficiency studies can be further research goals. The results can be beneficial for studying the resource efficiency impacts of various SIEM components and other architecture solutions for computing systems.

Author Contributions: Conceptualization S.M. and E.N.; methodology S.M.; software D.I.; validation E.N. and D.I.; formal analysis S.M. and D.I.; data curation S.M. and D.I.; writing—original draft preparation S.M. and E.N.; writing—review and editing D.I.; visualization S.M.; project administration, S.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are openly available in “Resource Efficiency of SIEM Components in a Virtual Environment” at http://dx.doi.org/10.17632/25v6shzfff.1.

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

Appendix A

The experiment was carried out with two different configurations of server software. In the first case, only the initial data are saved (see examples of data records in Appendix B, Figures A3 and A4). In the second configuration, a program code is added (Appendix B, Figure A6) that logs each request received by the server software. In this case, an additional record is created in the database for each received request, the general view of which is shown in Figure A5. These records are created after each POST request is executed.

The collection of data on the resources used is carried out using the atop utility at an interval of 1 s.

Sending requests are carried out in 4 threads, up to 10 simultaneous requests. The delay between sending packets of 10 requests is 300 ms, and the delay between stages of the experiment is 60 s. The maximum waiting for a response from the server is 10 s.
The code is executed using Node.js version 12.x.

The server software is launched under the control of the PM2 process manager with the parameters indicated in Table A1:

**Table A1. Server software launch parameters.**

| Parameter         | Value                                      |
|-------------------|--------------------------------------------|
| –node-args        | “--max_old_space_size = 1024”              |
| –i                | max                                        |
| –restart-delay    | 5                                          |
| –max-restarts     | 1000                                       |

This means that the amount of RAM allocated for the process does not exceed 1024 MB, the number of parallel processes is equal to the number of CPU cores (i.e., 2), the delay before restarting the process in case of failure is 5 s, and the maximum number of restart processes is 1000 times.

MongoDB version 4.2 is installed. The contents of the main configuration file (mongod.conf) are shown in Figure A1.

```plaintext
storage:
  dbPath /var/lib/mongodb
  journal:
    enabled: true

systemLog:
  destination: file
  logAppend: true
  path /var/log/mongodb/mongod.log

net:
  port: 27017
  bindip: 0.0.0.0

processManagement:
  timeZoneInfo /usr/share/zoneinfo
```

**Figure A1. MongoDB configuration.**

The sequence diagram for the experiment to assess the resource efficiency of recording user actions in the event log is shown in Figure A2.

After creating a VM, installing and running the server software and DBMS, the experiment itself begins. The initial data is loaded into the RAM of the client VM. After they have been fully loaded, sending data with the specified parameters begins.

In the first stage of the experiment, POST requests are sent to the server to save the ResearchSubject records. In the second stage of the experiment, POST requests are sent to the server to save the ResearchResult records. In both cases, each POST request contains information about only one record. Thus, the number of requests corresponds to the number of original data records.
Appendix B

The ResearchSubject documents (Figure A3) have the following attributes:

1. _id—unique document identifier.
2. sessionId—session identifier serving for client-server interaction.
3. login—pre-generated user login.
4. researcherId—identifier serving as foreign key for other document collection.
5. alias—user login substitute.
6. privateResearchSampleId—identifier serving as foreign key for other document collection.
7. createdAt—date and time when the action log entry was created.
8. updatedAt—date and time when the action log entry was updated for the last time.

Figure A2. Experiment sequence diagram.
9. privateResearchResults—array of ResearchResult documents.

```json
{
    "_id": "5d4c953e695df37f98ce5d1f17",
    "researchId": "fCbfEjWbrb1CHQy52QWEHJJuQh88Z6AqVp59p9QWIEE5E6N6n2r62OBYnTDoW9H",
    "login": "abchcf123441198007",
    "researcherId": "5d4b31d3f3037f1f19b03bbf",
    "alias": "abchhhhh",
    "privateResearchSampleId": "5d4e93ed9e60a57b7f423aaaa",
    "createdAt": "2019-11-28T10:44:56.127Z",
    "privateResearchResultId": []
}
```

**Figure A3.** Example of a ResearchSubject document.

The ResearchResult documents (Figure A4) have the following attributes:

1. _id—unique document identifier.
2. embeddedPsychotestId—identifier serving as foreign key for other document collection.
3. embeddedPsychotestId—order number of the document.
4. data—JSON object of various structure.
5. researcherId—identifier serving as foreign key for other document collection.
6. privateResearchSampleId—identifier serving as foreign key for other document collection.
7. privateResearchSubjectId—identifier serving as foreign key for other document collection.
8. createdAt—date and time when the action log entry was created.
9. updatedAt—most recent date and time when the action log entry was updated.

```json
{
    "_id": "5d4b31d3f3037f1f19b03bbf",
    "embeddedPsychotestId": "5d4e93ed9e60a57b7f423aaaa",
    "embeddedPsychotestId": 4,
    "data": {
        "answers": [
            // Multiple Objects
        ],
        "scales": []
            // Multiple Objects
    },
    "metadata": {
        "platform": "Windows 10 64-bit",
        "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3290.110 Safari/537.36 Edge/18.17763",
        "platformDescription": "Microsoft Edge 18.17763 on Windows 10 64-bit",
        "startTime": "2019-11-23T10:44:56.127Z",
        "endTime": "2019-11-23T10:44:56.127Z"
    }
}
```

**Figure A4.** Example of a ResearchResult document.

The ActionLog documents (Figure A5) have the following attributes:

1. _id—unique document identifier.
2. userId—unique identifier marking the user, which has executed the remote method.
3. exists—Boolean flag, marking if the user is present in the database at the moment of action logging.
4. request—name of the remote method.
5. createdAt—date and time when the action log entry was created.
6. updatedAt—most recent date and time when the action log entry was updated.

```json
{}
```

Figure A5. Example of an ActionLog document.

The hook, which is triggered after the received request is completed, is presented in Figure A6. It is written in JavaScript and its source code can be read as follows.

For the specified remote method, after the method code is executed:

1. Extract HTTP—request body data.
2. Pick id and researchSubjectId attributes from the data.
3. Set user identifier as researchSubjectId or id if the researchSubjectId is not present.
4. If there is no user identifier, then prevent the following code from being executed. It is expected behavior, as such HTTP requests are not processed using the remote method due to access control policies.
5. If there is a user identifier, then find the corresponding document in the database and then save an action log entry, containing data, presented in Figure A5.

```javascript
Model.afterRemote('**', (ctx, modelInstance) => {
  const params = ctx.req.body.data || ctx.req.body;
  const { id, researchSubjectId } = params;
  const userId = researchSubjectId || id;
  if (!userId) {
    return Promise.resolve();
  }
  return Model.app.models.ResearchSubject.findById(userId)
    .then((researchSubject) => {
      Model.app.models.ActionLog.logEvent(modelInstance, ctx, userId, researchSubject);
    });
});
```

Figure A6. Source code for logging requests for server software.

The monitoring results documents (Figure A7) have the following attributes:

1. client—object, representing the results of monitoring the resources of the client VM.
2. server—object, representing the results of monitoring the resources of the server VM.
3. mongodb—object, representing the results of monitoring the resources of the DBMS VM.

These objects contain key–value pairs, where the key contains the time in Unix time, and the value is a set of objects that reflect the consumption of VM resources at that time. The structure of the objects with monitoring results for a specified second is self-explanatory, and the recorded data correspond to the documentation of the atop utility (https://linux.die.net/man/1/atop, accessed on 27 February 2021).
References

1. Sancho, J.C.; Caro, A.; Ávila, M.; Bravo, A. New approach for threat classification and security risk estimations based on security event management. Future Gener. Comput. Syst. 2020, 113, 488–505. [CrossRef]

2. Miloslavskaya, N. Designing blockchain-based SIEM 3.0 system. Inf. Comput. Secur. 2018, 26, 491–512. [CrossRef]

3. Coppolino, L.; D’Antonio, S.; Mazzeo, G.; Romano, L. Cloud security: Emerging threats and current solutions. Comput. Electr. Eng. 2017, 59, 126–140. [CrossRef]

4. Al-Duwairi, B.; Al-Kahla, W.; AlRefai, M.A.; Abdelqader, Y.; Rawash, A.; Fahmawi, R. SIEM-based detection and mitigation of IoT-botnet DDoS attacks. Int. J. Electr. Comput. Eng. 2020, 10, 2182–2191. [CrossRef]

5. Kim, H.; Ben-Othman, J.; Mokdad, L.; Son, J.; Li, C. Research Challenges and Security Threats to AI-Driven 5G Virtual Emotion Technologies. Inf. Comput. Secur. 2020, 317–327. [CrossRef]

6. Ali, R.F.; Dominic, P.D.D.; Ali, S.E.A.; Rehman, M.; Sohail, A. Information Security Behavior and Information Security Policy Compliance: A Systematic Literature Review for Identifying the Transformation Process from Noncompliance to Compliance. Appl. Sci. 2021, 11, 3383. [CrossRef]

7. Machin, J.; Batista, E.; Martinez-Ballesté, A.; Solanas, A. Privacy and Security in Cognitive Cities: A Systematic Review. Appl. Sci. 2021, 11, 4471. [CrossRef]

8. Torres, N.; Pinto, P.; Lopes, S.I. Security Vulnerabilities in LPWANs—An Attack Vector Analysis for the IoT Ecosystem. Appl. Sci. 2021, 11, 3176. [CrossRef]

9. Gusev, A.; Ilin, D.; Gusev, A. Technology stack selection model for software design of digital platforms. Mathematics 2021, 9, 308. [CrossRef]

10. Magomedov, S.G.; Kolyasnikov, P.V.; Nikulchev, E.V. Development of technology for controlling access to digital portals and platforms based on estimates of user reaction time built into the interface. Russ. Technol. J. 2020, 8, 34–46. [CrossRef]

11. Kim, H.; Ben-Othman, J.; Cho, S.; Mokdad, L. A Framework for IoT-Enabled Virtual Emotion Detection in Advanced Smart Cities. IEEE Netw. 2019, 33, 142–148. [CrossRef]

12. Dilawari, A.; Khan, M.U.G.; Al-Otaibi, Y.D.; Rehman, Z.-U.; Rahman, A.-U.; Nam, Y. Natural Language Description of Videos for Smart Surveillance. Appl. Sci. 2021, 11, 3730. [CrossRef]

13. Ali, R.F.; Dominic, P.D.D.; Ali, S.E.A.; Rehman, M.; Sohail, A. Information Security Behavior and Information Security Policy Compliance: A Systematic Literature Review for Identifying the Transformation Process from Noncompliance to Compliance. Appl. Sci. 2021, 11, 3383. [CrossRef]

14. Torres, N.; Pinto, P.; Lopes, S.I. Security Vulnerabilities in LPWANs—An Attack Vector Analysis for the IoT Ecosystem. Appl. Sci. 2021, 11, 3176. [CrossRef]

15. Nikulchev, E.; Ilin, D.; Gusev, A. Technology stack selection model for software design of digital platforms. Mathematics 2021, 9, 308. [CrossRef]

16. Gusev, A.; Ilin, D.; Nikulchev, E. The dataset of software components experimental evaluation for application design selection directed with the artificial bee colony algorithm. Data 2020, 5, 59. [CrossRef]

17. Magomedov, S.; Lebedev, A. Protected Network Architecture for Ensuring Consistency of Medical Data through Validation of User Behavior and DICOM Archive Integrity. Appl. Sci. 2021, 11, 2072. [CrossRef]

18. Pun tambe kar, A.A. Analysis and Design of Algorithms: Conceptual Approach; Technical Publications: Pune, India, 2020.