Corporate chat under DLP–system controlling

Larisa Cherckesova[1], *Olga* Safaryan[1*], *Irina* Reshetnikova[1], *Tatyana* Nikishina[1], and *Denis* Korochentsev[1]

[1] Don State Technical University, Gagarin Square, 1, Rostov-on-Don, 344003, Russia

Abstract. The article describes the preparation process of software complex (package) consisting of secure corporate chat and DLP (Data Leak Prevention) system in the form of security monitor. Software development, project architecture, testing and other creation stage are demonstrated. The operation processes of security monitor are shown; the functionality of software package DLP–system is considered.

1 Introduction

At present, when most areas of people's lives, systems and industrial enterprises are computerized, the issue of ensuring the information security in such systems is actual. There are both external and internal threats, but for several decades, computer science and technologies has been developing rapidly and successfully precisely in the direction of creating antivirus systems and information protection systems from external intruders, not considering security incidents caused by internal violators (intruders) to be significant. According to Interstate Standard GOST R 53114–2008, information security incident is understood as any unforeseen or undesirable event that may disrupt the activity or information security [1-3]. Although the industrial enterprises use various software and hardware systems and complexes to protect information; programs for cryptographic transformation of stored data, which protects assets and other valuable information from hacker attacks, it is difficult to exclude the influence of the human factor. It can include unintentional mistakes of employees, operators and administrators, as well as deliberate conspiracies, transfer of information to competitors, and espionage. In this regard, the problem of information protection at the industrial enterprise or another company from the internal threats becomes extremely relevant.

The purpose of the article is implementation of the complex of software tools consisting of secure corporate chat and DLP (data leak prevention) system to protect the data from internal information leaks in any enterprise, especially in the small business.

This software package is corporate chat under control of DLP–system – is intended for use by employees inside the organization for the purpose of setting and performing their work tasks, as well as communicating within these tasks and exchanging files [4].

The role model of access control and DLP subsystem, the functionality of which is to detect the possible internal threats and to prevent them, ensures the security and safety of information inside the corporate chat.

* Corresponding author: safari_2006@mail.ru

The following functional requirements for software complex are defined:
- Sending the messages and the files in the corporate chat and e–mail client;
- Creation / deletion / edition of corporate chat objects (users, departments, tasks, roles);
- Control of the employee correspondence in the corporate chat;
- Tracking the distribution path of files transmitted in the chat and e–mail;
- Threat detection in the automatic and the manual modes;
- Saving of information into the database;
- Saving system events to the log;
- Alert the security officer about all incidents.

Scheme of interaction between the software components is shown in the Fig. 1.

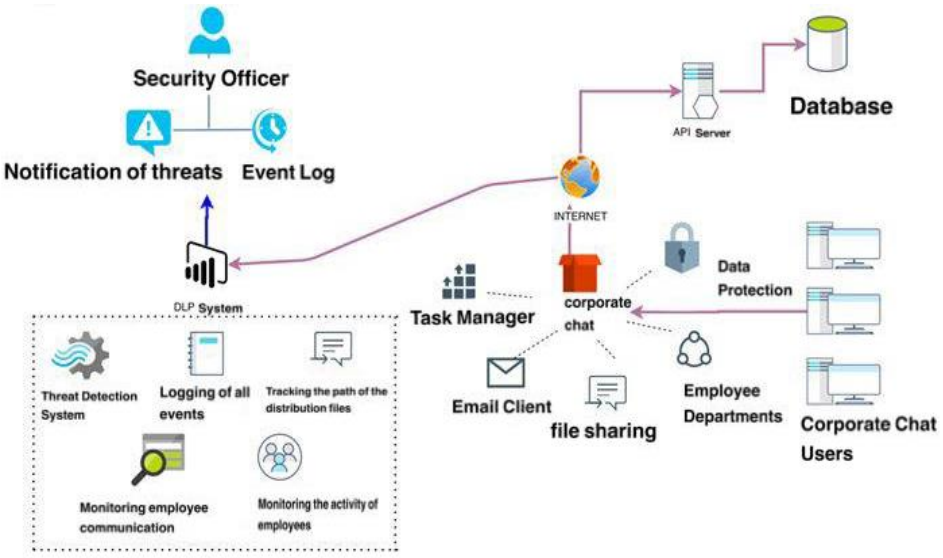

Fig. 1. The scheme of the components interaction of the software package.

2 Mathematical model and solution method

Software operation requires the several components: the corporate chat, shared server, and DLP system represented by security monitor.

Corporate chat has client – server architecture based on the thin client principle, that is, most of the data processing will be computed on the server [5].

Within the framework of this software implementation, the development of the software tool is aimed at small business enterprises, so there is no need to process the data on users' personal computers to reduce the load on the server.

At the user authorization level, the role identifier ID must be defined, according to which, after successful login attempt, users will be divided into 2 non–equal groups: the administrators and the regular users. The scheme of the chat architecture based on the role–based access control model is shown in the Fig. 2.

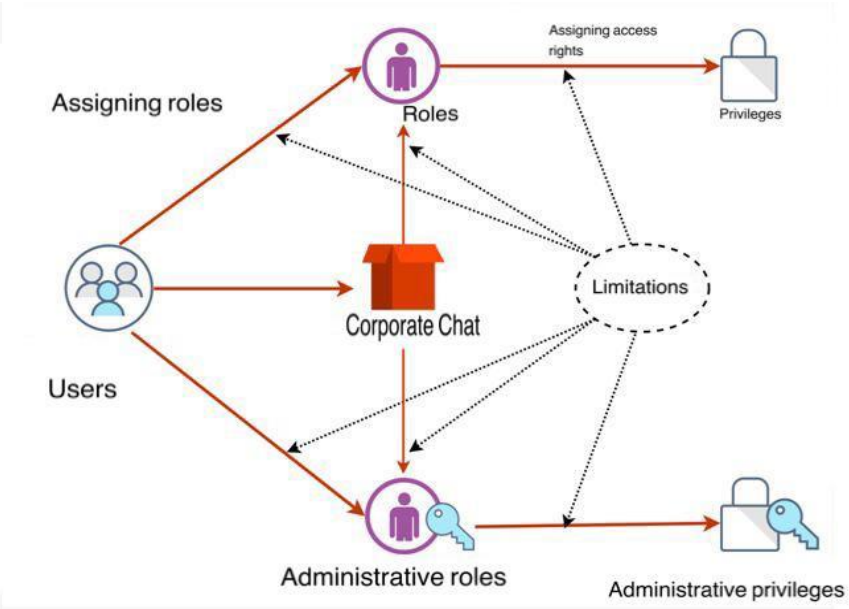

Fig. 2. Scheme of chat architecture based on the role–based access control model.

Looking at the chat in more detail, the following main algorithms should be highlighted: user authentication in the software; creation / edition / deletion of the objects; and the sending of messages and files in the corporate chat and the e–mail client of the chat. The users' authorization scheme is shown in the Fig. 3.

The user enters his username and password. The system identifies them based on the data stored in the database, and if it is successful, the user logs in the program and the event is recorded in the system. If the user enters incorrect data three times, the account is blocked and incident report is sent to the security officer or administrator, who, in turn, investigates the incident and decides to unblock the account or no.

During authentication, the role IDs are checking in the system, and then the list of permissions is loaded. Depending on this data, the program interface appears – windows, menus, tabs. For example, if the user has the right to create the tasks, then the tasks menu will have the "Create" tab and vice versa [6].

The security monitor consists of algorithms for searching for prohibited phrases in the dictionary in the corporate chat, detecting the internal threats in automatic and manual modes, logging the events and notifying the administrator about violations.

Threat detection algorithm detects violations based on the following signatures:

1. If there is forbidden word or expression in the chat, threat with low priority is detected.

2. If employees of different departments start the dialogue or send files to each other without any reason – when there is no task assigned to perform between them – this situation means that there is high level of threat.

3. If employees send files or e–mails to external e–mail addresses that do not correspond to their corporate e–mail addresses – there is high level of threat.

4. The number of attempts to log users into the system is recorded, and if there are more than 3 of them, the account is blocked, and this situation has high level of threat.

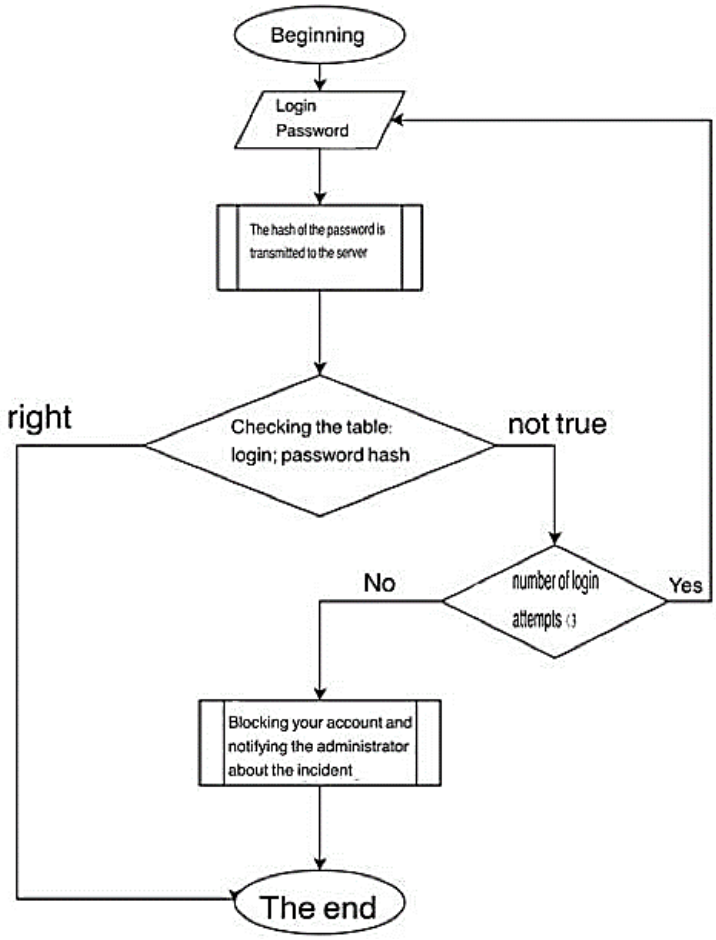

Fig. 3. User's authorization scheme.

The dictionary of forbidden words or phrases can be quite extensive, with different contexts. However, if it finds such phrase, the security officer decides manually, since full automation of the system requires long and deep research, beyond the scope of this work, the text analyzing with connecting neural networks and sentiment analysis phrases.

3 Results of research

The core of software is CRM class, which is responsible for creating the lists of the system subjects and objects – roles, users, tasks, departments, and dialogs. This class is also responsible for graphical navigation through the application – dividing the viewport into the list and the detailed one, and for the server initializing and loading all the program components. CRM class contains methods such as loadTask() – for loading the current task list, updateDialogList () – for updating the dialogs list, PreInit () – for server initialization and loading of main windows available to user according to his privileges.

Fig. 4 shows the code snippet responsible for initializing the main windows of the application. The current user is determined, then the ID of his role is defined, and if the user

does not have privileges to create new user, task, user data, departments, etc., then these buttons are removed from his menu.

```
public partial class MsgDetail : Page
{
    string msgList = "";
    DispatcherTimer timer;
    Dialog dialog;
    public MsgDetail(CrmEntity dialog)
    {
        InitializeComponent();
        title.Content = dialog.getTitle();


        this.dialog = (Dialog) dialog;
        subtitle.Content = this.dialog.getPositionUser();
        timer = new DispatcherTimer();
        timer.Tick += new EventHandler(timer_Tick);
        timer.Interval = new TimeSpan(0, 0, 1);
        timer.Start();

        updateDialog();
    }
```

Fig. 4. Code snippet responsible for initializing and updating the dialog.

Three key API classes are implemented for interacting with the server: API, TaskAPI, and UserAPI. The API class has the set of methods LoadTasks, LoadMsg, LoadDialog, and LoadDepartament for loading the objects of types Rules, Dialog, Department, and Event from the server and translating them into the objects of the specified type from the string in JSON format. Loading the data about departments and translating of the information from the database into JSON format is shown in Fig. 5.

```
public static void loadDepartament()
{
    Dictionary<int, Departament> departaments = new Dictionary<int, Departament>();
    using (WebClient wc = new WebClient())
    {
        var json = wc.DownloadString(API.urls["DEPARTAMENT_LIST"]);

        var usersJson = JArray.Parse(json);
        foreach (var u in usersJson)
        {
            departaments.Add(int.Parse(u["idDepart"].ToString()), new Departament(
                u["NameDepart"].ToString(), int.Parse(u["idDepart"].ToString())

                ));
        }
    }
    CRM.departaments = departaments;
}
```

Fig. 5. Code snippet responsible for the departments uploading.

The list of all server nodes is shown in the Fig. 6.

```
public static void initClassicServer()
{
    API.urls.Clear();
    API.urls.Add("USER_LOGIN", "http://dlp.profitapp.pro/api/user/login/");

    API.urls.Add("TASK_GET", "http://dlp.profitapp.pro/api/task/get/");
    API.urls.Add("TASK_ADD", "http://dlp.profitapp.pro/api/task/add/");
    API.urls.Add("TASK_UPDATE", "http://dlp.profitapp.pro/api/task/update/");
    API.urls.Add("TASK_COMPLETE", "http://dlp.profitapp.pro/api/task/complete/");
    API.urls.Add("TASK_WORK", "http://dlp.profitapp.pro/api/task/work/");
    API.urls.Add("TASK_STOP", "http://dlp.profitapp.pro/api/task/stop/");

    API.urls.Add("MSG_LIST", "http://dlp.profitapp.pro/api/msg/list/");
    API.urls.Add("MSG_GET", "http://dlp.profitapp.pro/api/msg/get/");
    API.urls.Add("MSG_SEND", "http://dlp.profitapp.pro/api/msg/send/");


    API.urls.Add("ROLE_ADD", "http://dlp.profitapp.pro/api/role/add/");
    API.urls.Add("ROLE_UPDATE", "http://dlp.profitapp.pro/api/role/update/");

    API.urls.Add("DEPARTAMENT_ADD", "http://dlp.profitapp.pro/api/departament/add/");
    API.urls.Add("DEPARTAMENT_UPDATE", "http://dlp.profitapp.pro/api/departament/update/");
    API.urls.Add("DEPARTAMENT_LIST", "http://dlp.profitapp.pro/api/departament/get/");

    API.urls.Add("USER_ADD", "http://dlp.profitapp.pro/api/user/add/");
    API.urls.Add("USER_UPDATE", "http://dlp.profitapp.pro/api/user/update/");

    API.urls.Add("EVENT_ADD", "http://dlp.profitapp.pro/api/event/addClient/");
}
```

Fig. 6. Lists of server nodes.

This software implementation does not use the dedicated server, but virtual one, and the database is located locally on the same computer. HTTP and POST requests are used to interaction and communication with server [7]. The code snippet that demonstrates the formation of the POST request is shown in the Fig. 7.

```
public static string POST(String url, String param)
{
    String result = "";
    WebRequest request = WebRequest.Create(url);
    request.Method = "POST";
    byte[] byteArray = System.Text.Encoding.UTF8.GetBytes(param);
    request.ContentType = "application/x-www-form-urlencoded";
    request.ContentLength = byteArray.Length;
```

Fig. 7. Fragment of the code of POST request formation.

For sending the requests to the server, Update () method is also used, which uses API class methods such as POST () and GetParamsString () to form the queue of URL requests, as shown in the Fig. 8.

```
public static Boolean Update(string query, Dictionary<string, string> paramsQuery)
{
    Console.WriteLine(API.GetParamsString(paramsQuery));
    var responceJson = JObject.Parse(
        POST(API.urls[query], API.GetParamsString(paramsQuery))
    );
    Console.WriteLine(responceJson["query"].ToString());
    return responceJson["status"].ToString() == "1";
}
```

Fig. 8. Snippet of Code Update().

Back to navigation in the application: its workspace is divided into 2 areas.

On the left side, there is the list block, which is used to navigate through the system objects – this is where we need to use the CrmEntity interface, since it allows using one list template for the entire system, and it is represented by the ListItem class [8].

On the right side, there is the window for detailed overview of the elements. This block is used within CRM for objects reviewing and editing. To edit CRM objects, the special forms have been developed that use the methods of the API class to transfer operations to the server. There are also the number of windows that only allow viewing objects, without the ability to edit them [9].

DLP–system security monitor has Web–interface written in programming language JavaScript, and Angular framework was chosen [8, 9]. This framework provides more opportunities to organize efficiently Web–projects and manage their logics and layout [10].

The monitor structure consists of 2 types of files: HTML files contain layout, and ts–files contain the logics. The analogue of class API from desktop app "Corporate chat" was implemented – class dlpService, communicating with server using the necessary API requests, such as getUserById(), getUsers(), getEvents(), getIncident(), getTasks(), getRules(), getDepartaments(), getMsgs() and getChats() [11].

The method findIncident() checks the list of all system events and by certain signature detects the threat and determines its level.

The ChatComponents class loads the dialog between 2 users in the security monitor and displays messages using the methods ngOnInit(), getUser (), and toChat ().

The EventComponents class displays user groups by department on the page and displays the events of the EventDlp type on the page. The getUser() method outputs the user IDs. The getObj() method determines the type of event in the system (figure 9).

The IncidentComponent class determines incidents from the list of system events and puts them in the "incidents" tab. It contains the similar getObj () method that detects incidents such as "Password Brute force", "Unsubstantiated dialog", "Potentially dangerous phrases", "File output to external address" [12].

The layout of each page is written in HTML, includes the page for viewing dialogues, messages between users, the page for viewing of information about employees by department, and the page with events and incidents [13].

The developed software tool consists of 3 components: corporate chat, DLP– system security monitor, and server [14]. The interaction between the components takes place via HTTP protocols. The MySQL database is used as the storage subsystem. The software tool was developed using the C#, PHP, and JavaScript programming languages with the Angular framework. For the components of software complex, the algorithmic design was carried out with application of schemes, and the description of main classes and methods.

```
public getObj(type:string,e:EventDlp ){
  switch(type){
    case "-":{
      return "-"
      break;
    }
      case "user" :{
        return this.getUser(e.detail).fio
      }
      case "login" :{
        return e.detail
      }
      case "dialog_start" :{
        e.level = 0
        if(this.getUser(e.second_user_id).idDepartament == this.getUser(e.user_id.toString()).idDepartament) e.level = 1
        return `${this.getUser(e.second_user_id).fio} [${e.second_user_id}]`
      }
      case "file_send_dialog" :{
        console.log(e);
      //   if(this.getUser(e.second_user_id).idDepartament == this.getUser(e.user_id.toString()).idDepartament) e.level = 2
        return `${this.getUser(e.second_user_id).fio} [${e.second_user_id}]`
      }
      case "file_send_mail" :{
        e.level = 2
        return `${e.detail} [внешний]`
      }
      case "try_login_free" :{
        return `${e.detail}`
```

Fig. 9. The method getObj() of the EventComponents class.

4 Discussion and conclusions

The article describes the implemented software complex consisting of secure corporate chat and DLP–system in the form of security monitor; defines the functional requirements, and performs the algorithmic and the software design [15]. Its advantages include speed performance, effectiveness, simplicity of using, controlling of the employees correspondence with the search of forbidden words and phrases in the dictionaries; the sufficient minimal functionality for tracking the events on the system and detecting the threats, the prevention of the information leakage; and low cost software [16, 17].

References

1. GOST R 53114–2008 (Interstate Standard). Information security. Ensuring information security in the organization. Basic terms and definitions.
2. O. Safaryan, E. Pinevich, E. Roshchina, et al., E3S Web of Conferences, **224**, 01041 (2020)
3. V.V. Zhilin, I.I. Drozdova, I.A. Sakharov, et al., in Proceedings of IEEE East-West Design and Test Symposium, EWDTS 2019, Institute of Electrical and Electronics Engineers Inc., 8884375 (2019), DOI: 10.1109/EWDTS.2019.8884375
4. DLP–Systems. Data Protection with DLP–system. What is DLP and How They Work, URL: https:// searchinform.ru/informatsionnaya–bezopasnost/dlp–sistemy/ (accessed: 19.10.2020).
5. Client–Server Application Architecture, URL: https://training.qatestlab.com/blog/ technical-articles/ client–server–architecture/ (accessed: 21.10.2020).

6. Features. Advantages and Benefits of Using C# in Applications, URL: https://cyberleninka.ru/article/n/osobennosti–dostoinstva–i–preimuschestva–ispolzovaniya–s–v–prilozheniyah (accessed: 01.11.2020).
7. Advantages of PHP, URL: http://www.php.su/php/?option (accessed: 02.11.2020).
8. HTTP and Server Interaction. HttpClient and Sending Requests, URL: https://metanit.com/ web/angular2/6.1.php (date accessed: 21.11.2020).
9. Ten Advantages of Using the Framework Angular.js when Developing Web–Applications (2020), URL: https://stfalcon.com/ru/blog/post/why-use-angularjs-for-webapps
10. N. Okumura, K. Ogata, Y. Shinoda, Journal of Information Security and Applications, 53, 102529 (2020)
11. Y. Wang, K.S. Cheng, M. Song, E. Tilevich, Science of Computer Programming, **181**, 27–46 (2019)
12. J. Benymol, A. Material Story: Proceedings, **24(3)**, 2036–2040 (2020)
13. E. Thouvenin, G. Schoehn, F. Rey, et al., Journal of Molecular Biology (JMB), **307**, 161–172 (2001)
14. C. Tong, Q. Wang, Y. Gao, et al., Electric Power Systems Research, **113**, 228–236 (2014)
15. D. Winder, Info Security, 7(2), 38–41 (2010)
16. S. Prichard, Info Security, 8(4), 18 – 21 (2011)
17. F.M. Faiz, J.A.M. Alazab, A. Shalaginov, Future Generation Computer Systems, **110**, 744 – 757 (2020)