Clustering context items into user trust levels

Paweł Lubomski[1] and Henryk Krawczyk[2]

[1] IT Services Centre, Gdańsk University of Technology, Gdańsk, Poland
lubomski@pg.gda.pl
[2] Faculty of Electronics, Telecommunications and Informatics,
Gdańsk University of Technology, Gdańsk, Poland
hkrawk@eti.pg.gda.pl

Abstract. An innovative trust-based security model for Internet systems is proposed. The TCoRBAC model operates on user profiles built on the history of user with system interaction in conjunction with multi-dimensional context information. There is proposed a method of transforming the high number of possible context value variants into several user trust levels. The transformation implements Hierarchical Agglomerative Clustering strategy. Based on the user's current trust level there are extra security mechanisms fired, or not. This approach allows you to reduce the negative effects on the system performance introduced by the security layer without any noticeable decrease in the system security level. There are also some results of such an analysis made on the Gdańsk University of Technology central system discussed.

Keywords: context, system security, clustering, TCoRBAC model, trust levels.

1 Introduction

Internet systems are the most commonly used types of systems nowadays. Their popularity results from the ease of their use. They are accessible from nearly all types of computers – from traditional PCs to the more and more popular mobile devices, such as tablets and smartphones. It does not matter what operating system is used, because they are used with web browsers which implement the same standards of HTML, JavaScript, AJAX, etc. They are also accessible from any place where the Internet is available. So nearly everyone can try to access them. On the other hand, such openness poses a serious threat to their security. Also, their distributed and service-oriented architecture has a significant impact on their security and reliability.

The technology evolves very quickly. Every five years there is a major step made – a small revolution. The same applies to the area of security. There were solutions based on traditional MAC (Mandatory Access Control), DAC (Discretionary Access Control) and RBAC (Role Based Access Control) models [1], [2]. Then, they were expanded by using elements of dynamicity – they adaptively expand and shrink users' permission sets [3]. Finally, the analysis of widely understood context was involved [4]–[7]. The expansion of the idea of "big data" created a new place for security mechanisms. They started to profile users' behavior, and decide upon access on the

basis of these profiles [8]–[10]. In such a way we evaluate from strictly defined access control rules to user trust-based mechanisms – the decision of granting or denying access is based on the system-to-user trust [4], [11]–[13].

There is also another important matter connected with the security of Internet systems. There should be a balance between the security level and the usability and efficiency of the system [14]. From the system point of view, the security mechanisms should provide an acceptable level of risk, but from the user's point of view the security layer should be nearly invisible as long as the user behaves in a predictable way [15].

There are two main problems in building such efficient security mechanisms. The first one is the complexity of context and its analysis. The context nature is multidimensional. That results in high complexity. Previously we proposed the universal CoRBAC (Context-oriented Role Based Access Control) model and the security audit-based procedure of determining the system security level [6], [16], [17]. The other problem is that strong security mechanisms have a negative impact on the system performance. The remedy for this problem is building some correct and efficient user trust estimation procedures and forcing strong security mechanisms only on low user trust levels. This is the subject of this paper.

2 User trust model for Internet systems

Users behave in a predictable way. The same applies to their interaction with systems. For example, they usually use the same devices, the interaction takes place in the same hours and from the same place. This is very unique to each user and can be treated as the user's fingerprint. As mentioned earlier, the user interaction with the system is accompanied by many context parameters which describe the current overall context of each activity. A written log of such activities, with accompanying contexts, creates an individual user profile. On the basis of such a profile, the system can determine if the user behaves typically, or not. That determines the system-to-user trust levels. We proposed the model with different security mechanisms which depend on the user trust level [13]. Now we want to deal with the problem of determining the correct amount of user trust levels.

Let us define context (C) as a set of context parameters (CP_i):

$$c_x \in C = CP_1 \times CP_2 \times \ldots \times CP_z \tag{1}$$

Each context parameter (CP_i) is a finite, discrete set of possible values of the parameter:

$$CP_i = \{cp_{i1}, cp_{i2}, \ldots, cp_{in}\} \tag{2}$$

Some context parameters have a continuous character (e.g. time). During the context analysis they are clustered into certain groups, e.g. days of the week. The same situation applies to context parameters with discrete but numerous values, e.g. set of IP addresses. They are clustered into subsets of a specific netmask.

The current context (c_x) accompanying each user with the system interaction is described by the current values of each context parameter (CP_i). Thus, the cardinality of the set of context items is the multiplication of cardinality of each context parameter values set.

Let us analyze an example of a context consisting of three context parameters:

- CP_1 – logical localization of user – a set of the following elements: cp_{11} = internal network, cp_{12} = campus network, cp_{13} = external network (Internet),
- CP_2 – time of user's activity in the system – a set of elements: cp_{21} = weekday, cp_{22} = weekend,
- CP_3 – type of device being used – a set of elements: cp_{31} = traditional PC, cp_{32} = mobile device.

Then the $|C| = 12$. Each context is a three-value tuple, e.g. $c_1 = (cp_{11}, cp_{21}, cp_{31})$. When any of the context parameter changes the value, the current context also changes. For example when the user changes the type of the device used, the accompanying context also changes.

We can create a bi-directional full graph of items transitions – see fig. 1. The vertexes of this graph are context items (c_i). The edges of the graph describe the changes of the context accompanying user requests. A history of such a user with system interaction with accompanying context value creates a user's profile. Each of the vertexes are also marked by a number determining how often the user interacted in this context (see fig. 2a). We can cluster vertexes of this graph into groups on the basis of this metric (*f*). The resulting groups form user trust levels, on the basis of which the appropriate security mechanisms are fired.

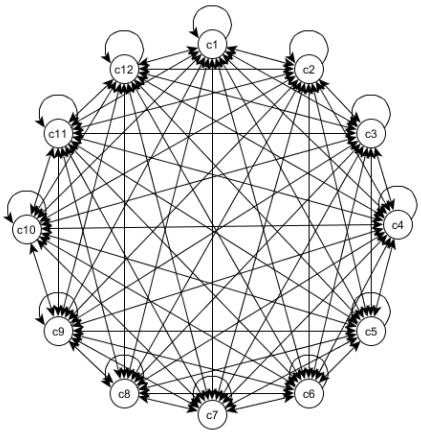

Fig. 1. Example bi-directional graph of possible user context items transition

It is worth mentioning some facts. Firstly, the multiplicity of context parameters results in time-consuming procedures of determining their values. Secondly, the num-

ber of items in the graph (context items) is very numerous and grows exponentially. Thus, it is hard to analyze such a big amount of data, which again has a negative impact on the system performance.

3 Classification algorithm for determining categories in user trust model

Each user profile can be represented as a histogram of the previously mentioned frequency metric for each context value. Let us analyze the example graph of context items presented in fig. 2a. Each vertex has a frequency metric *f* assigned which determines how often the user interacted in this context. An example histogram has been depictured in fig. 2b. Of course, in the real world the context is much more complicated, multi-dimensional (consists of many more context parameters), and the context parameters have more different values. So the effective context items set is very numerous.

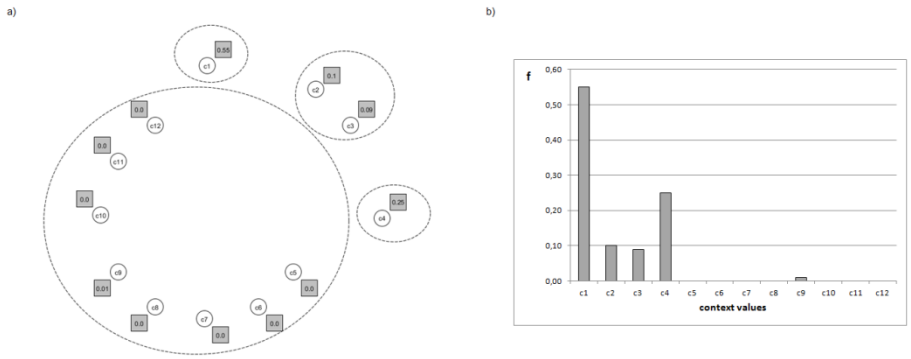

Fig. 2. a) Vertexes are marked by a number determining how often the user interacted in this context and clustered into 4 groups **b)** Histogram of frequency metric *f* which determines how often the user interacted in the specific context values

There is a need to cluster the frequencies from the histogram into groups. For these groups some appropriate trust levels are assigned. Each trust level is associated with a corresponding security mechanism. The security mechanisms may include the following: retyping captcha-code, entering the user password again, inputting a special code sent by SMS or e-mail, etc.

The best way to determine the number of the clusters, and to do the clustering, is by using Hierarchical Agglomerative Clustering (HAC) [18]–[20]. As an input for this algorithm there is a vector of frequency metrics from the user's profile used. Fig. 3a presents an example dendrogram as a result of such clustering (all HAC analysis were made with the use of P. Wessa software [21]). There is the Euclidean distance metric used, and the Centroid Criterion strategy of computing the distance between

the clusters. Fig. 3b presents example distribution of the number of clusters and the distance between the closest clusters (also known as the height of dendrogram).

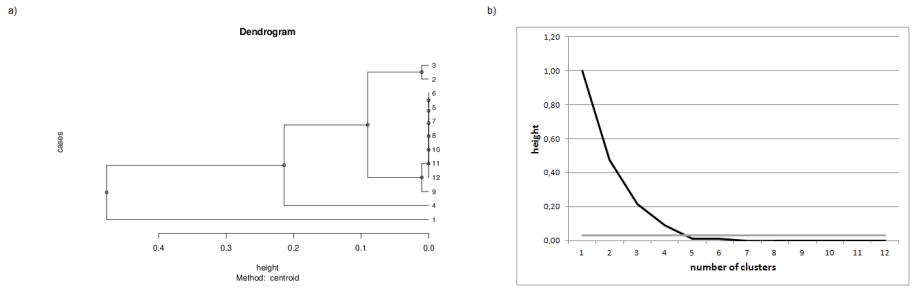

Fig. 3. a) Example dendrogram as a result of HAC **b)** Example distribution of the number of clusters and the distance between the closest clusters (height)

On the basis of this distribution it can be determined when the distance stops to decrease significantly – this is the most optimal number of clusters. After this point the differences between clusters are so tiny that it is pointless to extract any additional trust levels, and corresponding security mechanisms. Of course, this number is also dependant on the characteristics of the system, and the number of reasonable security mechanisms that may be introduced.

The final step is to determine the ranges of frequency metrics corresponding with each cluster, sort them ascending, and assign them trust levels with security mechanisms. This way we reduce the number of items (context values) from very numerous to four, in our case. This number results from the analysis of our university central system logs.

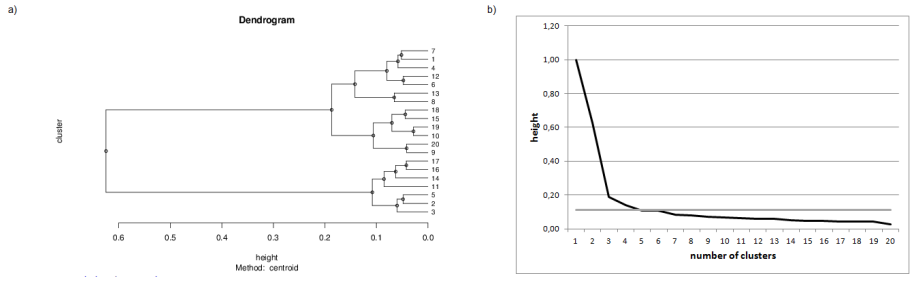

Fig. 4. a) Dendrogram and **b)** distribution of the number of clusters and the distance between the closest clusters (height) in our university case

Fig. 4 presents the dendrogram and distribution of the number of clusters and the distance between the closest clusters in the case of our university system. The analysis is taken on a population of 3,000 users. The results are presented for the top 20 clusters cut – the subsequent clusters have small distances (near zero) and can be omitted,

they would result only in the illegibility of the charts. The system is used by over 40,000 users. The analysis takes into account a nearly four-year period of the system's operation.

There is another approach – with the use of unsupervised K-means strategy [22]. The initial number of clusters may be determined using a Mean shift algorithm [23]. Next, there should be the results of determining the average distance from the centroid for each cluster (treated as imperfection of clustering) compared. The analysis should start with the number of clusters determined by the Mean shift algorithm. Then, there should be the nearby numbers of clusters checked too. Finally, the best number of clusters is when the imperfection metric stops to decrease significantly (similarly to HAC). In our case this strategy gives some similar results to HAC strategy, but it is less deterministic. Thus, in our opinion, HAC strategy seems to be a better approach to items reduction and transformation to trust levels.

4 Application of trust model in TCoRBAC approach

The above trust level determination mechanism was applied to a TCoRBAC (Trust- and Context-oriented Role Based Access Control) model [13], [17]. The main goal of this approach is to build separate profiles of each user's interaction with the system. Using the strategy described in this paper of the items' reduction, and assigning appropriate user trust levels, it is possible to fire some extra security mechanisms. The algorithm of authorization and trust verification in the TCoRBAC model is presented as the pseudo-code below.

```
foreach (userReq) do:
  cx = determine current context;
  authResult = do the basic permission check;
  if(authResult == true)
  then
    profile = compute current profile of user(userReq);
    clusters = do the HAC clustering (profile);
    trustLvl = determine cluster containing current con-
               text cx of userReq;
    result = enforce extra security mechanism (trustLvl);
    if(result satisfies security policy)
    then grant access;
    else deny access;
  else deny access;
```

The result of clustering the example items transition graph (mentioned at the beginning of the paper) is depicted in fig. 5. There are four clustered items with the appropriate trust levels assigned. A user interacts in some context which changes while changing one of its context parameter values. This may cause a change of the trust level and adequate security mechanism.

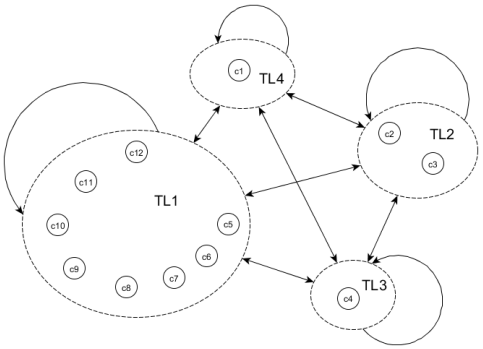

Fig. 5. Items transition graph as a result of clustering procedure and trust level assignment

The security mechanisms connected with trust levels may be as follows:

- TL4 – fully trusted user – there is no extra security action (cost: 0 second)
- TL3 – confirmation of user identity by pointing to 1 of 10 images (cost: 1 second)
- TL2 – re-entering the password or other personal data (cost: 3 seconds)
- TL1 – inputting the one-time-password sent by SMS (cost: 5 seconds)

Table 1 presents an example simple scenario of user interaction with the university system. There is the time spent on each action (from the user's point of view) indicated, as well as the type of action: business or security.

Table 1. Example simple scenario of user interaction with the university system

N^o	Activity	Time spent by user	Type of activity
1	Login	3 sec.	τ_S
2	Computation of current context	≈ 0 sec.	τ_S
3	Permission check	≈ 0 sec.	τ_S
4	Determination of trust level	≈ 0 sec.	τ_S
5	Execution of security mechanism	0 / 1 / 3 / 5 sec.	τ_S
6	"Student" application choice	1 sec.	τ_B
7	Selection of subject	1 sec.	τ_B
8	Grade reading	1 sec.	τ_B
9	Logout	1 sec.	τ_S

On the basis of this example scenario there is calculated the user-side efficiency metric (ε) defined as the ratio of the time spent on business activities (τ_B) to the overall time spent on business and security activities ($\tau_B + \tau_S$):

$$\varepsilon = \tau_B / (\tau_B + \tau_S) \tag{3}$$

The resulting efficiency metric varies from 0.25 to 0.4286, depending on which security mechanism is fired. It has a nearly double difference, which is noticeable for

the user. Of course, in more complex scenarios this metric will decrease. It is worth mentioning that users mostly interact in TL4, so then the efficiency metric is from the top of the range. The metric decreases in lower trust levels, but it is the result of a higher security level.

The TCoRBAC model was developed for the central system of Gdańsk University of Technology. More information about this model can be found at [6], [13], [16], [17].

5 Conclusions

It is worth pointing out that the described approach is very effective in complicated multi-dimensional cases. The process of clustering is not very efficient, but it may be done asynchronously in the background. There may be one more optimization done: the clustering and profile computation may be actualized periodically – this also reduces the load of the system.

The reduction of many context items to a few trust levels does not decrease the system security level significantly and does not have an impact on the user-side efficiency metric (ε). With reference to Furnell's insights [14], the security level is nearly invisible for the users who do not change their behavior – they are classified at the highest trust level, so there is no need to do any extra security checks.

On the other hand, when the user changes the context very often, s/he gets a regular distribution of frequency metric *f*, so that it is hard to cluster their profile. In such a case it is worth considering doing global (per whole system) clustering and determining global ranges of this metric which cannot be exceeded during per-user clustering and trust level assigning.

The clustering process may also take into account a weight metric pointing to the distance between the context values (computed on the basis of how many context parameter values change from one context value to another). It is a quite intuitive approach but also causes some blur of clusters and decrease of the system security level. In such a case an intruder may behave quite similarly to the real user (use similar context values) and will not be detected. If we use strict context clustering (with equal weight = 1) the intruder has to behave exactly the same as the user (the same context values) in order not to be detected. In the real world, where many context parameters are taken into account, it is very hard to copy the user's behavior precisely.

References

1. M. Benantar, Access Control Systems. Security, Identity Management and Trust Models. Springer-Verlag, 2006.
2. E. Bertino, "RBAC models — concepts and trends," Computers & Security, vol. 22, no. 6, pp. 511–514, Sep. 2003.

3. A. Ricci, M. Viroli, and A. Omicini, “An RBAC Approach for Securing Access Control in a MAS Coordination Infrastructure,” in 1st International Workshop “Safety and Security in MultiAgent Systems” (SASEMAS 2004), 2004, pp. 110–124.
4. R. Bhatti, E. Bertino, and A. Ghafoor, “A Trust-Based Context-Aware Access Control Model for Web-Services,” Distributed and Parallel Databases, vol. 18, no. 1, pp. 83–105, Jul. 2005.
5. M. F. F. Khan and K. Sakamura, “Context-aware access control for clinical information systems,” in 2012 International Conference on Innovations in Information Technology (IIT), 2012, pp. 123–128.
6. H. Krawczyk and P. Lubomski, “CoRBAC – context-oriented security model (in Polish),” Studia Informatica, vol. 34, no. 3, pp. 185–194, 2013.
7. X. Huang, H. Wang, Z. Chen, and J. Lin, “A Context, Rule and Role-Based Access Control Model In Enterprise Pervasive Computing Environment,” in 2006 First International Symposium on Pervasive Computing and Applications, 2006, pp. 497–502.
8. M. Miettinen and N. Asokan, “Towards security policy decisions based on context profiling,” in Proceedings of the 3rd ACM workshop on Artificial intelligence and security - AISec ’10, 2010, p. 19.
9. A. Gupta, M. Miettinen, N. Asokan, and M. Nagy, “Intuitive Security Policy Configuration in Mobile Devices Using Context Profiling” in 2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing, 2012, pp. 471–480.
10. C. Manikopoulos and S. Papavassiliou, “Network intrusion and fault detection: A statistical anomaly approach,” IEEE Communications Magazine, vol. 40, no. October, pp. 76–82, 2002.
11. S. De Capitani Di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, G. Psaila, and P. Samarati, “Integrating trust management and access control in data-intensive Web applications,” ACM Transactions on the Web, vol. 6, no. 2, pp. 1–43, May 2012.
12. J. W. Woo, M. J. Hwang, C. G. Lee, and H. Y. Youn, “Dynamic Role-Based Access Control with Trust-Satisfaction and Reputation for Multi-agent System,” 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops, pp. 1121–1126, 2010.
13. H. Krawczyk and P. Lubomski, “User Trust Levels and Their Impact on System Security and Usability,” in Communications in Computer and Information Science, Springer International Publishing, 2015, pp. 82–91.
14. S. Furnell, “Usability versus complexity – striking the balance in end-user security,” Network Security, vol. 2010, no. 12, pp. 13–17, Dec. 2010.
15. S. P. S. Pahnila, M. S. M. Siponen, and A. M. A. Mahmood, “Employees’ Behavior towards IS Security Policy Compliance,” 2007 40th Annual Hawaii International Conference on System Sciences (HICSS’07), 2007.
16. P. Lubomski, “Context in Security of Distributed e-Service Environments,” in Proceedings of the Chip to Cloud Security Forum 2014, 2014, p. 18.
17. P. Lubomski and H. Krawczyk, “Practical evaluation of security mechanisms of Internet systems (in review),” IEEE Security & Privacy Magazine.
18. R. P. Adams, “Hierarchical Agglomerative Clustering.” 2016.
19. S. P. Borgatti, “How to Explain Hierarchical Clustering,” Connections, vol. 17, no. 2, pp. 78–80, 1994.
20. A. Bouguettaya, Q. Yu, X. Liu, X. Zhou, and A. Song, “Efficient agglomerative hierarchical clustering,” Expert Systems with Applications, vol. 42, no. 5, pp. 2785–2797, 2015.

21. P. Wessa, “Free Statistics Software, Office for Research Development and Education, version 1.1.23-r7,” 2016. [Online]. Available: http://www.wessa.net/.
22. J. a. Hartigan and M. a. Wong, “A K-Means Clustering Algorithm”, Journal of the Royal Statistical Society, vol. 28, no. 1, pp. 100–108, 1979.
23. D. Comaniciu and P. Meer, “Mean shift analysis and applications” in Proceedings of the Seventh IEEE International Conference on Computer Vision, 1999, pp. 1197–1203 vol.2.