

# ENHANCING THE WORDPRESS SYSTEM: FROM ROLE TO ATTRIBUTE-BASED ACCESS CONTROL

Lifeng Cao, Jia Ying Ou, Amirhossein Chinaei

EECS Department, York University, Toronto, Canada

## **ABSTRACT**

*Role-Based Access Control (RBAC) is the most commonly used model on web applications. The advantages of RBAC are the ease of understanding, applying and managing privileges. The static RBAC model cannot alter access permission in real-time without human involvement and therefore the model suffers from increasing false negative (and/or false positive) outcomes. Hence, the Attribute-Based Access Control (ABAC) model has been proposed to introduce dynamicity and minimize human involvement in order to enhance security. WordPress is a very popular Role-Based content management system. To our best knowledge, no solution to merge from RBAC to ABAC model for WordPress applications has been found. Our contribution is a WordPress plug-in that we have developed to build ABAC upon the existing RBAC setups. In this journey, we have investigated various scenarios by studying different application categories to come up with an enhanced automatic model that adds real-time grant and revoke feature to WordPress.*

## **KEYWORDS**

*Role-Base-Access-Control, Attribute-Base-Access-Control, WordPress, Content Management, Security*

## **1. INTRODUCTION**

WordPress is a popular free and open source Content Management System (CMS) that uses a legacy Role Based Access Control (RBAC) model to administer the privileges. It is essential to prevent unauthorized users to obtain confidential information and perform privilege escalation that may commit to cybercrime.

How popular is WordPress? Thirty-three percent of the entire internet uses WordPress (cf. Table 1), it is around 19,500,000 websites [1]. WordPress is used for content listing and blogging. It is known by its ease of use, versatility, themes, and the use of plugins. It is the quickest growing CMS, and it is translated into 40 different languages. There are approximately 500 new sites that are built in the top 10 million websites on the internet every day [1]. In addition, there are 17 posts that are published every second on WordPress sites globally. The word “WordPress” is being google searched 37 million times each month [1] (cf. Figure 1).

The access control component of WordPress is based on the RBAC model. It assigns permissions to each user who is connected to the network based on their organizational role assignments [2]. The limitation on RBAC is that user’s permission depends on the role of the user only, which is a lack of flexibility. It is a time-consuming process for administrators to create new roles for a certain employee to do a certain task and delete the role after the task is completed. This process often needs to be done in real-time. It also leads to the problem of role explosion when large

Dhinaharan Nagamalai et al. (Eds) : WIMO, ICAIT, ICDIPV, NC, CRYPIS - 2019  
pp. 107-122, 2019. © CS & IT-CSCP 2019 DOI: 10.5121/csit.2019.90410

amounts of roles are being created [3]. This is where Attributed-Based Access Role (ABAC) [4,5] model comes into play. ABAC has been identified to defeat these limitations of RBAC [6,7,8]. ABAC is a model that assigns policies to each user based on the existing attributes of the user instead of the user's role. Therefore, it is a more adequate solution to handle various factor of situations by using attributes like *user role*, *location*, *time*, *administration configuration*, and so on [6]. It also offers more flexibility and portability in comparison to RBAC [6,5], which makes it a more suitable access control across organizational domains.

For the already widespread use of WordPress and the trend that many organizations would benefit by adapting the ABAC model [9,7], a plug-in automatically merges RBAC to ABAC seems to be the best solution. A role-based only approach is not wise for data privacy since selecting a role that is too general can lead to potential breaches, and having a lot of specific roles causes a role explosion. For the growing concerns on the limitation of RBAC [6], our proposed approach lies in an attribute-based solution. Our goal is to devise and integrate comprehensive mechanisms towards decentralized privacy-preserved administration models with web-based and implemented by application plug-ins.

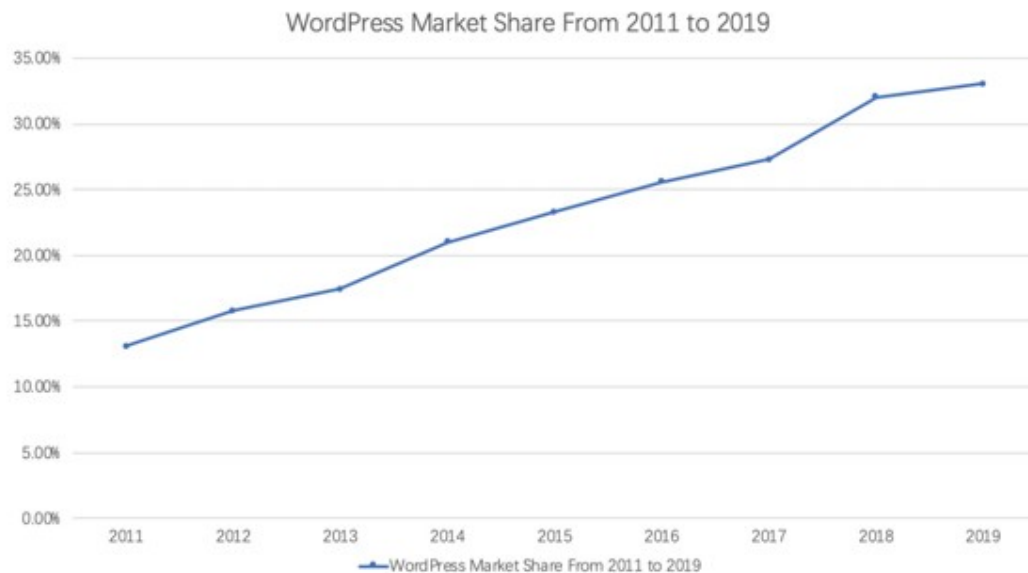


Table 1. [10] WordPress' Overall Market Share

The main contribution of our work is the design and development of a plug-in for the WordPress application, by studying various application categories. Any existing client of WordPress can use this plug-in to automatically merge from RBAC to ABAC. The assignment of permissions are based on user's attributes that are retrieved from employer's employment database. The plug-in enhances flexibility and security; and, it redefines the role capabilities for each user as needed. The solution that we propose in this paper follows these steps: 1) Layout the roles within the department structure of each organization. 2) Integrate the organization roles to WordPress roles, which is the RBAC model. 3) Merge the RBAC model into ABAC. The solution involves transforming roles in an organization to user attributes [7] in the ABAC model and adds attributes according to database information to automatically grant privileges at the run time together with time constraints-administered by the grantor or system.

We build our solution based on this observation that the ABAC attributes can be put into five categories [11]. Those are user attributes, object attributes, environmental attributes, connection

attributes, and administrative attributes. User attributes are attributes of the subjects of the system. Object attributes are the attributes of the resources of the system. Environmental Attributes are attributes derived from the current state of the system's environment. Connection Attributes that only apply to the current session of use. And administrative attributes are configuration attributes that apply to the whole system and are either manually set by an administrator or by some automated process. In our plug-in, user attributes, object attributes, and environmental attributes are the core ones the solution is based on.

The rest of this paper is organized as follows: Section 2 outlines the previous work and how our work contributes to it. In Section 3, our analysis of the three cases is studied. In Section 4, our approach to merging WordPress application from RBAC to ABAC is presented. In Section 5, a prototype of WordPress plug-in is introduced. Lastly, in Section 6, the benefits of our proposal and future directions are concluded.

## 2. LITERATURE REVIEW

Various ways of combining RBAC and ABAC models to maintain the advantages and limit the disadvantages have been researched broadly. Kuhn et al [12] proposed a NIST initiative that has introduced three ways to combine the benefits of ABAC and RBAC model. They are Dynamic Roles, Attribute Centric and Role Centric. However, this paper did not provide a comprehensive explanation of the schemes.

For Dynamic Role, Kern et al [13] and Al-Kahtani et al [14] has introduced the solution by using attribute-based rules for automated user-role assignments in large companies to overcome the difficulties in reviewing permission assignments while evaluating the effects of new rules or modification of rules in dynamic role-assignment. However, these solutions only for user attributes, and it does not provide a solution for the role-explosion problem of RBAC. In contrast to our approach, we provide a solution to resolving the role-explosion problem by transferring the user role from RBAC to user attributes in ABAC.

For Role Centric, Jin et al [15] proposed a Role-Centric Attribute-Based Access Control Model (RABAC), this model is an integration of roles and attributes using role-centric methodology. RABAC model extends RBAC with permission filtering policy (PFP). PEP puts constraints to the set of permission base on attributes. This is a useful model in addressing the role-explosion problem, and it is easy for user-role assignment. However, this model is not applicable to systems that change attributes frequently. Our approach is suitable for frequently attribute changing systems.

Rajpoot et al [3] proposed the novel way to maintain the advantages of both RBAC and ABAC model by combining the two models together. In their approach, contextual information was put into consideration during the access control decision-making process. This approach limits the need for creating a large number of roles by providing a fine-grained access control mechanism. In this model, audit for permission requests is simple because it still uses the concept of role-centric. It is comparatively straightforward to view the effects of adding or removing a policy because the policy specification is at the level or role.

Huang et al [16] proposed a model that has two levels: underground and aboveground levels. The aboveground level is RBAC model that is extended with environmental constraints. And the underground level utilizes attribute-based policies to assign user-role and role-permission process. Even though this proposed model uses fine-grained access control, the massive number of role creating causes role explosion problem. Since attribute-based policy is used for user-role, role-permission assignment, it is not simple for auditing and policy visualization. This proposed

framework applies the attribute-based policies during the process of establishing available permissions, not afterward, which is different from the paper [3] and paper [15].

Fernandez et al [17] introduced a new strategy on Access Control called: Category-Based Model for ABAC. A similar model called: Dynamic Event-Based model designed by Bertolissi et al [18] also shares this idea. By classifying entities to enhance ABAC into the following classes: categories, principals, actions, identifiers, answers and situation identifiers. Predefined actions are used to generate connections between each entity that is in different categories. Old ABAC system's policy is translated and against new generated Control based model. Flexibility and extensibility were increased by this model. But simultaneously, the complexity of implementation has significantly increased. These attributes make reliable, convenient and non-destructive transfer between RBAC to Category-Based Model impossible. In the method that this paper proposed, the easy-installed plug-in can enhance the original WordPress RBAC model to ABAC by a single click.

### 3. CASE STUDY

In the default WordPress application, there are six roles: Super Administrator, Administrator, Editor, Author, Contributor, and Subscriber. Figure 1 below demonstrates the hierarchy structure of roles in WordPress application [19], the higher the role in the hierarchy, the more permissions a role has. The super administrator has all the permissions that the administrator has; the administrator has all the permissions that editor has; the editor has all the permissions that contributor has and contributor has all the permissions that subscriber has.

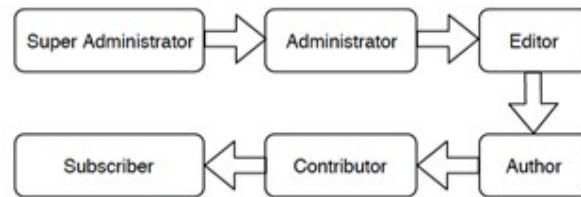


Figure 1. Hierarchy Structure of Roles in WordPress Application

Three case studies of translating department structure of a company to the RBAC model is illustrated in Section 3.1, 3.2 and 3.3. The information for department structure of the companies is extracted and concluded from the official career website of the company, third-party analytic website and company's employee profiles on LinkedIn. The following figures are a simplified version of our understanding of how these organizations use WordPress as their blogs and content listing websites. In addition, the problems of each company are compared and described in Section 3.4.

According to the traditional developing structure, large companies usually put the IT department into two different environments: Production and Non-Production environment. The production environment is the environment that is responsive to what customers can view, and the non-production environment is for internal developing and testing purpose, only internal employees can view, edit and delete. The reason for two different environments is to prevent sensitive information from leaking and maintain the good reputation of the company. Employees who have access to the non-production environment do not have access to the production environment necessarily.

In the following figures, all the roles are in different shapes. The rectangular shape indicates the role is working in both Non-Production (N-P) and Production (P) environment. The oval shape

indicates the role is working in the Non-Production(N-P) environment. The rectangular shape without two angles indicates the role is working in the Production(P) environment. And the triangle shape indicates the role comes from a different department. In all the b figures (2b, 3b, 4b), the extra word with underline is included to indicate the corresponding WordPress role.

### 3.1 Microsoft Blog Website Analysis

Figure 2a is an inverted tree that illustrates the Engineering Department Structure of Microsoft Company[20,21,22]. For example, Engineering Manager and Development Manager both have access in Non-Production and Production environment. Content Developer only has access to Non- Production environment. Marketing Department is a separated department from the Engineering department. And all other roles have permissions in the production environment.

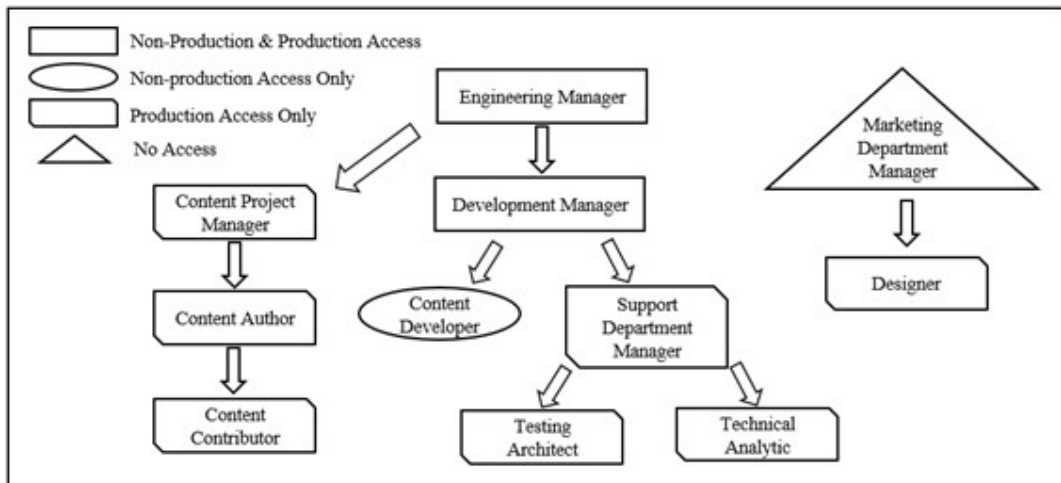


Figure 2a. Microsoft Department Structure

Figure 2b illustrates the integration of Microsoft company roles to the WordPress application roles. As we can see, the corresponding role in WordPress for Engineering Manager is an administrator who has the highest authority role in single site WordPress application. In this example, only the engineering department manager has the capability to immigrate between the two environments. Therefore, the Engineering Manager needs to have all the capabilities. For other roles, the same way of reading the figure applies.

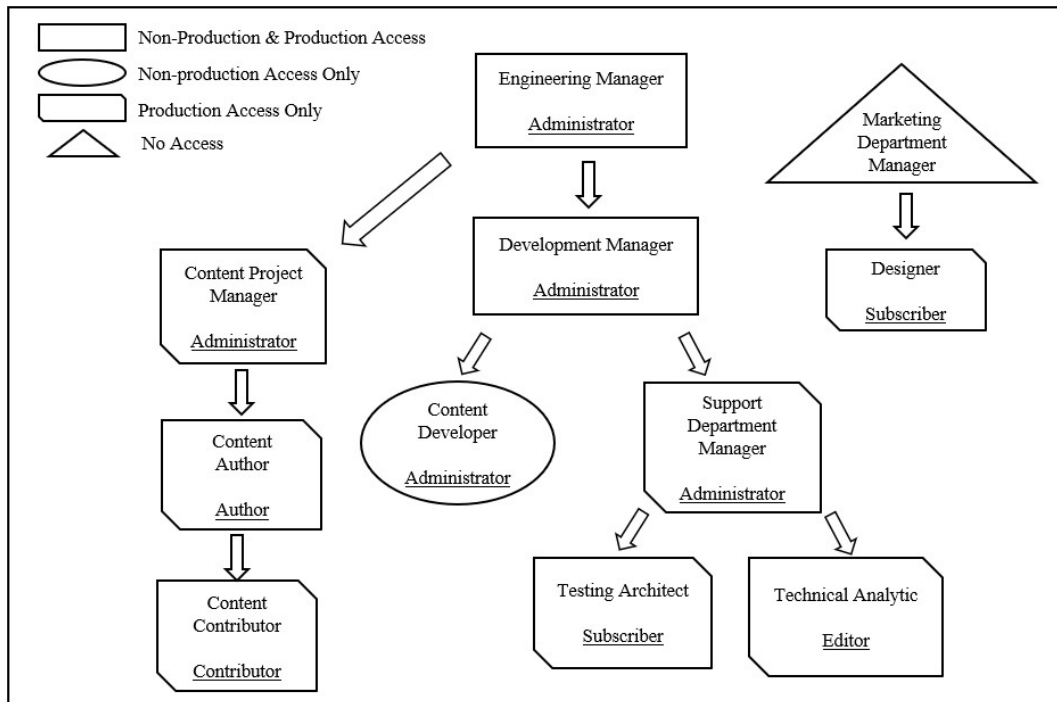


Figure 2b. Integration from Microsoft company role to WordPress role

### 3.2 Blackberry Blog Website Analysis

Figure 3a is an inverted tree that illustrates the Department Structure of Blackberry Company [23,24]. For example, Product Management Director and Product Development both have access in Non-Production and Production environment. Software Developer only has access to Non-Production environment. Marketing Department is a separated department from the Engineering department. And all other roles have permissions in the production environment.

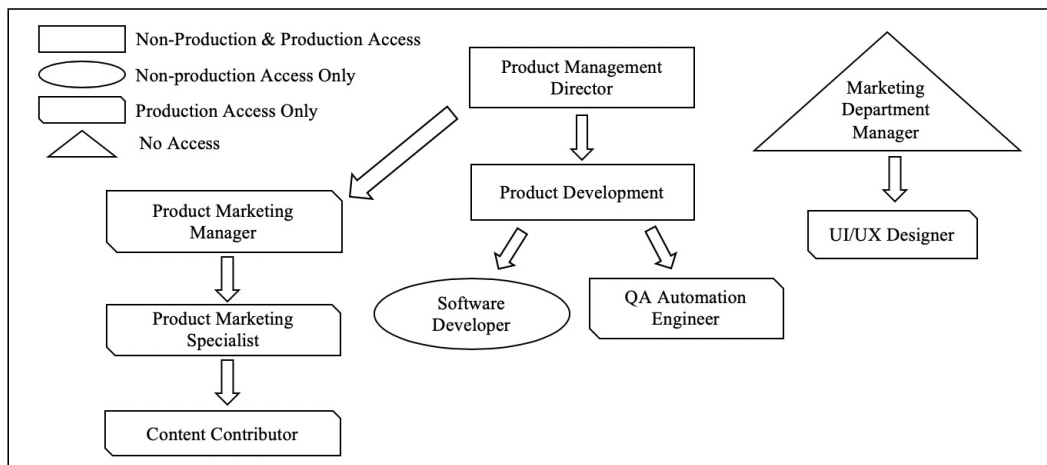


Figure 3a. Blackberry Department Structure

Figure 3b illustrates the integration of Blackberry company roles to the WordPress application role. BlackBerry blog is an installation of WordPress Multisite application. Product Management Director is assigned to the Super Administrator role to control WordPress net level settings.

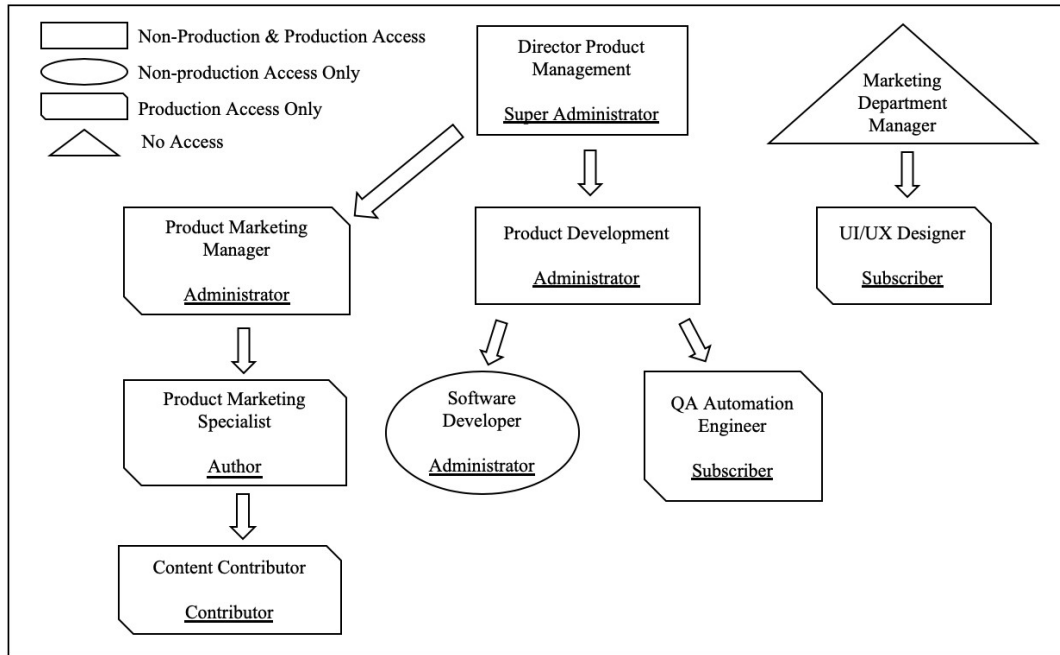


Figure 3b. Integration from Blackberry company role to WordPress Role

### 3.3 Bata Style Website Analysis

Figure 4a is an inverted tree that illustrates the Department Structure of Bata Style Websites [25]. For example, Department Manager and IT Manager have access in Non-Production and Production environment. Page Builder only has access to Non- Production environment. And all other roles have permissions in the production environment.

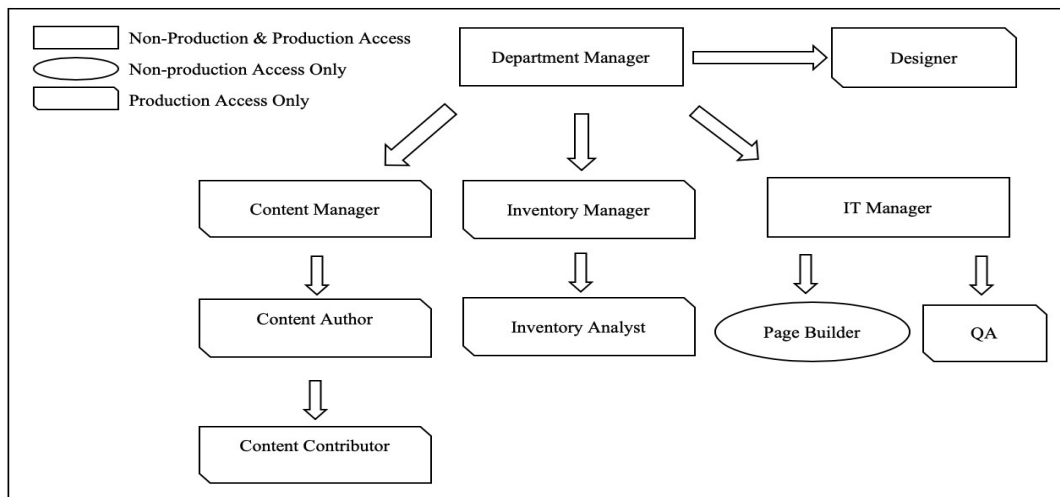


Figure 4a. Bata department Structure

Figure 4b illustrates the integration of Bata Style website roles to WordPress application role.

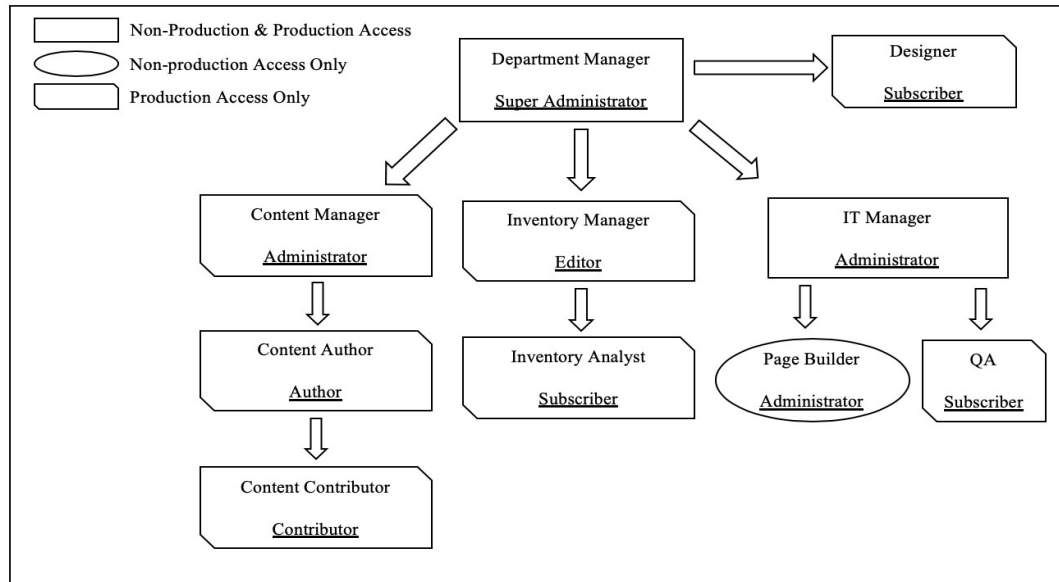


Figure 4b. Integration from Bata company role to WordPress Role

### 3.4 Problem Formulation

All three cases above have a few common issues when using the RBAC model. Firstly, it is extremely difficult and inefficient for large organizations to manage employee access, especially when it involves a contract, partner with third parties and modernize system [26]. For example, Microsoft is a complicated firm, and there are plenty of highly distributed projects to handle simultaneously. If the company assigns roles or shares accounts every time when a user needs additional permissions can lead to insecure of the system.

Secondly, with multisite WordPress application, Super Administrator has the highest permission on all the subsites. For BlackBerry, the owner of the Super Administrator account takes full responsibility on the whole website. The explosion of these accounts has significant security threats on the entire WordPress System.

Thirdly, the original WordPress system is not external user-friendly. For the Bata style website, it is time-consuming to switch roles for an external users for analysing company stocks. In addition, time constraints are extremely essential. To resolve these issues, this paper introduces a general solution against WordPress for all three cases.

## 4. Method

Before introducing the solution, here are the assumptions:

1. Database information is up to date
2. Employee Database contains - All employee's relationship.
  - Each employee's working location, working schedule.
  - Each employee's travel records (ongoing, pending)
  - Tasks records (ongoing, pending, and completed)



### 3. Action Audit

In our proposal, we categorized the default permissions of WordPress into two groups, sensitive and general. All the permissions that are responsive at the front end of the website are considered as sensitive permissions, general permissions otherwise. For sensitive permission, requests will be demonstrated in Section 4.1 by following the workflow in Figure 5. For general permission, requests will be demonstrated in Section 4.2 by following the workflow in Figure 6. We also describe an exceptional case in Section 4.3 where an employee requests permission beyond the scope of his/her job by following the workflow in Figure 7.

#### 4.1 Sensitive Permission Request

In this example, Joseph is an employee who holds a QA job position for the Bata Style website: [batastyle.com](http://batastyle.com). Employee Joseph is requesting for sensitive permission such as `delete_plugins`. Corresponding to Figure 5, a workflow on the ABAC system for requesting sensitive permission is illustrated. Once Employee Joseph requested the `delete_plugins` permission in Non-Production environment, the system checks the database record on whether the supervisor (IT Manager in this case) has assigned the task with the requested permission to the employee before.

Situation One:

1. IT Manager has the permission: `delete_plugins`.
2. IT Manager has assigned the tasks to Joseph before he requests permission
3. IT Manager is currently working or has logged into the system within the last 30 minutes.
4. Joseph and IT Manager are working at the same location currently.
5. Joseph grants the `delete_plugins` permission.

Situation Two:

1. IT Manager has the permission: `delete_plugins`
2. IT Manager has assigned the task to Joseph before he requests permission
3. The IT Manager is not currently working
4. The system will notify Joseph with IT Manager's schedule
5. Joseph needs to request sensitive permission again when the IT Manager is working
6. Joseph grants the `delete_plugins` permission.

Situation Three:

1. IT Manager did not assign the task to Joseph before he requests permission
2. Assigner has the requested permission
3. Notification will be sent to the assigner (IT Manager in this case)
4. After IT Manager assigns the task to Joseph, Joseph requests for `delete_plugins` permission again
5. Joseph grants the `delete_plugins` permission

Situation Four:

1. IT Manager did not assign the task to Joseph before he requests permission
2. Assigner does not have the requested permission
3. Permission will be denied

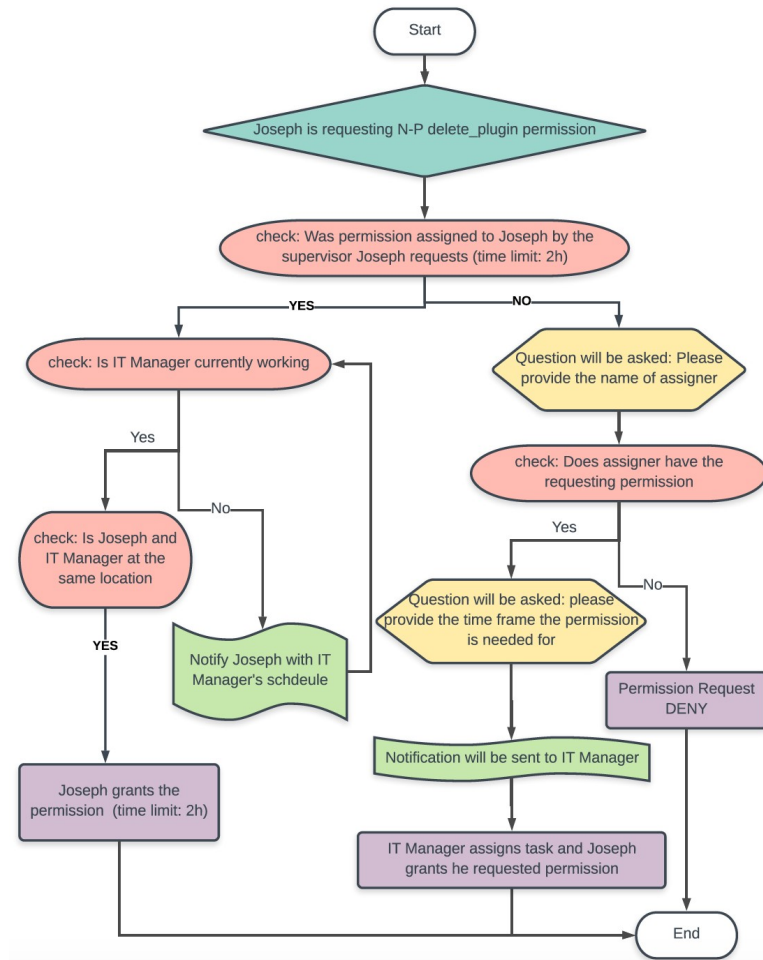


Figure 5. workflow on the ABAC system for requesting sensitive permission

## 4.2 General Permission Request

In this example, Emily is a designer at Microsoft company. Emily is requesting for general permission. Figure 6 demonstrates a workflow on the ABAC system for requesting general permission. Once Emily has requested the edit\_post permission, the system checks the database record on whether the supervisor (Department Manager in this case) has assigned the task with the requested permission to the employee before.

Situation One:

1. Department Manager has assigned the task Emily before she requests permission.
2. Emily grants permission.

Situation Two:

1. Department Manager did not assign the task to Emily before she requests permission.
2. Assigner has the requested permission.
3. Notification is sent to the Department Manager, and information is recorded for audit.

4. After Department Manager assigns the task to Emily, Emily requests the permission again.
5. Emily will then grant the permission.

Situation Three:

1. Department Manager did not assign the task to Emily before she requests permission.
2. Assigner does not have the requested permission.
3. Permission will be denied, information is recorded.

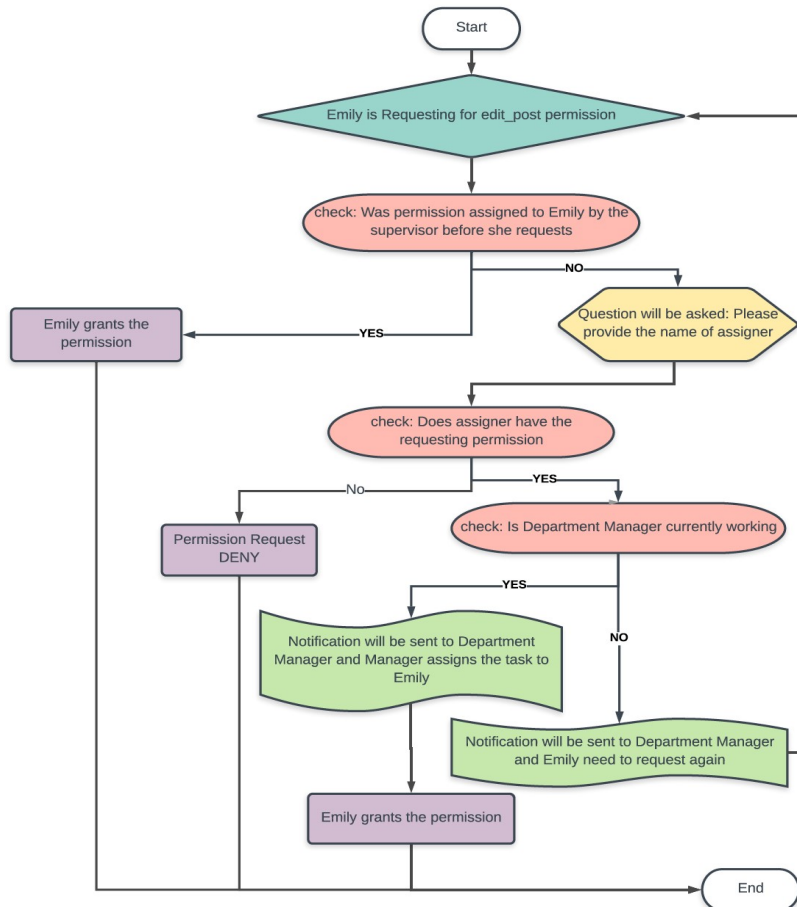


Figure 6. workflow on the ABAC system for requesting general permission

### 4.3 Out of Scope Permission Request

In this example, Olivia is an employee who holds a Content Contributor job position at Microsoft company. Olivia is requesting general permission. As we can see in Figure 7, it is a workflow on the ABAC system for requesting general permission. Once Olivia has requested the edit\_post permission, the system checks the database record on whether the supervisor (Content Author in this case) has assigned the task with the requested permission to the employee before. Content Author did not assign the task to Olivia before Olivia request permission. The person who assigned Olivia the task does not have the requested permission. Permission will be denied.

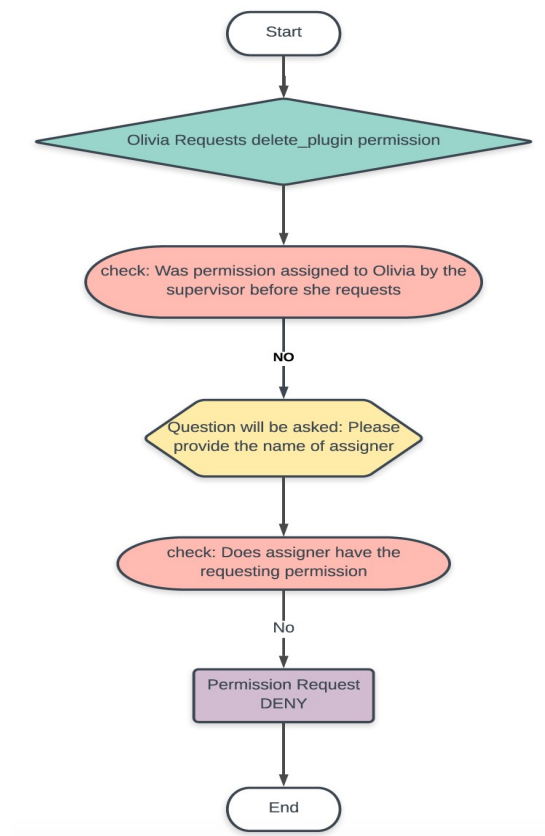


Figure 7. Workflow on the ABAC System for requesting an out of scope permission

## 5. PROPOSED IMPLEMENTATION

Based on section 4, a WordPress Plug-in Prototype implemented based on general WordPress Application. This prototype for WordPress Plug-in is named as Advanced Access Control. This plug-in helps users to assign permissions, request permissions and block suspicious requests based on user behaviours.

For the purpose of illustration, we provide a series of snapshots of this plug-in. By combining the internal `wp_users` table from WordPress database with the external employee database, the management chain is integrated into WordPress and used by the plug-in (cf. Figure 8). Supervisors can assign tasks with corresponding permissions to employees (cf. Figure 9) by going to Plug-in submenu that is called “Assign Tasks”. (cf. Figure 10)

ID	user_login	user_pass	user_nicename	user_email	supervisor	user_url	user_registered	user_activation_key	user_status	display_name
1	admin	SP\$BGu9D9raGka24znrBCIAKODGihZz31	admin	admin@test.com			2019-02-12 23:09:19		0	admin
5	Joseph	SP\$Ba2C8mnQTCdPCT08XVveQa8MXJYISE1	joseph	joseph@test.com			2019-03-04 21:08:01	1551733682.SP\$BHI0acJyLVss1we9V0Pw1Ag0tSnZ0	0	Joseph
6	Olivia	SP\$B3fp0LM1Z8lwT\$STw\$yOdZ3fy0e8.	olivia	olivia@test.com	Joseph		2019-03-04 21:08:35	1551733717.SP\$BQ454vowZZ7hJz6c.S5oBSQdnHLK3.	0	Olivia
7	Emily	SP\$BIZPJEioRjMIX76DpjQqzSTZN171f10	emily	emily@test.com	Olivia		2019-03-04 21:09:01	1551733743.SP\$BiqR17d6M6Rb65776S/pIERQ67U46.	0	Emily

Fig. 8: Data for `wp_data` table, Olivia (Editor) is supervisor of Emily (Author)

**You are supervising the following users**

Emily

**Please select the user you want to assign task to**

Emily

Select the person you want to assign task.

**Please select the permission you want to assign the person.**

Emily

edit\_others\_pages

Please change the header in my page.

Submit

(a)
(b)

Fig.9. Snapshot of plug-in. After selecting Emily as task receiver, Olivia is able to assign the permission and task to Emily. These information is stored in wp\_task table.

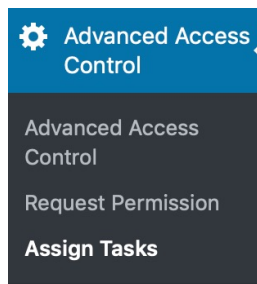


Figure 10. Plug-in Submenu - “Assign Tasks”

After task has been successfully assigned, email notification of the assigned task is sent to Emily. Emily is able to see this task in the submenu called “Request Permission”. (cf. Figure 11a)

⚙️ **Advanced Access Control**

Advanced Access Control

Request Permission

**Assign Tasks**

**You are having the following tasks.**

Your task ID is:5  
 Permission needed:edit\_others\_posts  
 Task detail:Please change the header in my page.  
 Task was assigned to you on:2019-03-04 17:14:01

Figure 11a: Snapshot of plug-in, Emily’s task

At this point, Emily do not have permission to edit other author’s post (cf. Figure 12b).

Title	Author	Categories	Tags		Date
<a href="#">Hello world!</a> <a href="#">View</a>	admin	Uncategorized	—	<span style="background-color: #ccc; border: 1px solid #ccc; border-radius: 50%; padding: 2px 5px;">1</span>	Published <u>2019/02/12</u>

Figure 12b: Do not have edit option on Hello World post which written by admin

If the permission request form submitted by Emily passed the following three checks as shown in Figure 13. Permission is granted to Emily for a fixed length of hours.

Figure 13 consists of three screenshots of a permission request form, labeled (a), (b), and (c). Each screenshot shows a form with three sections: 'Please select the permission you want to request.', 'Who assign you the task?', and 'How long do you need the permission?'. Below each section is a 'Submit' button.

- (a) Permission: `remove_users`; Assigner: `admin`; Duration: 2 hours.
- (b) Permission: `edit_others_posts`; Assigner: `Olivia`; Duration: 10000 hours.
- (c) Permission: `edit_others_posts`; Assigner: `Olivia`; Duration: 2 hours.

Figure 13: Snapshot of the plug-in, (a). no task was assigned to Emily that requires `remove_user` permission and none of the tasks was assigned by admin, permission request denied. (b). Emily should not request permission for an unreasonable length of time (c). test passed and permission is assigned to Emily for 2 hours.

After Emily has received the permission, she is able to edit admin's post for 2 hours. If she signs back in after 2 hours, permission is no longer available. (cf. Figure 14).

<input type="checkbox"/> Title	Author	Categories	Tags		Date
<input type="checkbox"/> Hello world! <a href="#">Edit</a> <a href="#">Quick Edit</a> <a href="#">View</a>	admin	Uncategorized	—		Published 2019/02/12
<input type="checkbox"/> Title	Author	Categories	Tags		Date

Figure 14: Emily is able to edit the post by clicking the “Edit” or “Quick Edit” button

## 6. CONCLUSIONS

In this paper, a novel extension from RBAC to ABAC model is proposed to take advantages of both models in the WordPress content management system. Our approach preserves the benefits of keeping the original roles for easy immigration from old to a new environment and maintaining the easiness of management. On the other hand, by using attribute-based policies, not only we avoid the role explosion problem, but we also minimize the need for human involvement in real time. This automation addresses the problem of false negative (and/or false positive) decisions that occur for unavailability of human at real-time. Hence, data security in WordPress is increased significantly. The installation of this plug-in is simple. Interested readers may download it from this link (<http://www.cse.yorku.ca/~ahchinaei/WP-plug-in>). Our future work will be introducing unsupervised learning on understanding third parties' policies and autogenerating additional policies for comprehensive permission constraint.

**REFERENCES**

- [1] K. K, "WordPress Stats: Your Ultimate List of WordPress Statistics (Data, Studies, Facts – Even the Little-Known)," codeinwp, 2019. [Online]. Available: <https://www.codeinwp.com/blog/wordpress-statistics/>.
- [2] D. Ferraiolo, R. Sandhu, S. L. Gavrilu and R. D. Kuhn, "Proposed NIST Standard for Role-Based Access Control," *ACM Transactions on Information and System Security*, vol. 4, pp. 224-274, 2001.
- [3] Q. M. Rajpoot, C. D. Jensen and R. Krishnan, "Attributes Enhanced Role-Based Access Control Model," *Trust, Privacy and Security in Digital Business*, pp. 3-17, 2015.
- [4] V. C. Hu, D. Ferraiolo, R. Kuhn, A. R. Friedman, A. J. Lang, M. M. Cogdell, A. Schnitzer, K. Sandlin, R. Miller and K. Scarfone, "Guide to Attribute Based Access Control (ABAC) Definition and Considerations(Draft)," *NIST Special Publication 800(2013)*, p. 162, 2013.
- [5] V. C. Hu, R. Kuhn, D. F. Ferraiolo and J. Voas, "Attribute-Based Access Control," *Artech House Information Security and Privacy*, vol. 48, no. 2, pp. 85-88, 2015.
- [6] X. Jin, R. Krishnan and R. Sandhu, "A Unified Attribute-Based Access Control Model Covering DAC, MAC AND RBAC," In *Data and Applications Security and Privacy XXVI*. Springer, pp. 41-55, 2012.
- [7] E. Coyne and T. R. Weil, "ABAC and RBAC: Scalable, Flexible, and Auditable Access Management," *IT Professional*, vol. 15, no. 3, pp. 14-16, 2013.
- [8] E. Yuan and J. Tong, "Attributed Based Access Control (ABAC) for Web Services," In: *International Conference on Web Services*, IEEE, 2005.
- [9] B. Gunjan, A. Vijayalakshmi, V. Jaideep and S. Shamik, "Enabling the Deployment of ABAC Policies in RBAC Systems," *Data Appl Secur Priv XXXII(2018)*, pp. 51-68, 2018.
- [10] "Market Share Statistics (2011-2019) Kinsta," Kinsta, 2019. [Online]. Available: <https://kinsta.com/wordpress-market-share/>.
- [11] D. Servos and S. L. Osborn, "Current Research and Open Problems in Attribute-Based Access Control," *Forthcoming article in ACM Computing Survey (CSUR)*, vol. 49, no. 4, p. Article No. 65, 2017.
- [12] R. D. Kuhn, E. J. Coyne and T. R. Weil, "Adding Attributes to Role-Based Access Control," *IEEE Computer*, vol. 43, no. 6, pp. 79-81, 2010.
- [13] A. Kern and C. Walhorn, "Rule Support for Role-Based Access Control," *10th ACM Symposium on Access Control Models and Technologies*, 2005.
- [14] M. A. Al-Kahtani and R. Sandhu, "A model for attribute-based user-role assignment," *18th Annual Computer Security Applications Conference*, 2002. *Proceedings.*, 2002.
- [15] X. Jin, R. Sandhu and R. Krishnan, "RABAC: Role-Centric Attribute-Based Access Control," In *Proceedings of the 6th International Conference on Mathematical Methods, Models and Architectures for Computer Network Security*. Springer, pp. 84-96, 2012.
- [16] J. Huang, D. M. Nicol, J. H. Huh and R. Bobba, "A Framework Integrating Attribute-Based Policies into Role-Based Access Control," *Proceedings of the 17th ACM symposium on Access Control Models and Technologies*, pp. 187-196, 2012.

- [17] M. Fernandez and B. M. Thuraisingham, "A Category-Based Model for ABAC," 18 Proceedings of the Third ACM Workshop on Attribute-Based Access Control, vol. 21, no. 3, pp. 32-34, 2018.
- [18] C. Bertolissi, M. Fernandez and S. Barker, "Dynamic event-based access control as term rewriting," Proceedings of the 21st annual IFIP WG 11.3 working conference on Data and applications security, pp. 195-210, 2007.
- [19] "Roles and Capabilities << WordPress Codex," WordPress.ORG, 2019. [Online]. Available: [https://codex.wordpress.org/Roles\\_and\\_Capabilities](https://codex.wordpress.org/Roles_and_Capabilities).
- [20] "Careers at Microsoft | Microsoft jobs," Microsoft, 2019. [Online]. Available: <https://careers.microsoft.com/us/en>.
- [21] "Microsoft | Departments," Research Gate, 2019. [Online]. Available: <https://www.researchgate.net/institution/Microsoft/departments>.
- [22] "Microsoft Jobs | LinkedIn," LinkedIn, 2019. [Online]. Available: <https://www.linkedin.com/company/microsoft/jobs/>.
- [23] "BlackBerry Ltd (BB.TO) People," Reuters, 2019. [Online]. Available: <https://www.reuters.com/finance/stocks/company-officers/BB.TO>.
- [24] "BlackBerry Jobs - BlackBerry Careers," BlackBerry, 2019. [Online]. Available: <https://ca.blackberry.com/company/careers>.
- [25] "Bata Salaries," Glassdoor, 2019. [Online]. Available: <https://www.glassdoor.ca/Salary/Bata-Salaries-E16097.htm>.
- [26] B. Fisher, N. Brickman, S. Jha, S. Weeks, T. Kolovos and P. Burden, "Attribute Based Access Control," In: NIST Special Publication 1800-3, 2017.