November 2020

A DYNAMIC ROLE SWITCHING APPROACH TO ENFORCE PRINCIPLE OF LEAST PRIVILEGE

HP INC

A DYNAMIC ROLE SWITCHING APPROACH TO ENFORCE PRINCIPLE OF LEAST PRIVILEGE

ABSTRACT

This document issues a method to consider adding a proxy module that can be dynamically switched to different granted roles in order to decouple different business functional requests and to be executed under the least permissions. This document designs this module as an independent component in the example system architecture, but this disclosure will not be limited by this architecture design.

BACKGROUND

Principle of least privilege (PoLP) is one of the most important security concept in modern system design, which is commonly implemented in most of operating systems. Role-Based Access Control (RBAC) is a well-known approach that supports PoLP, and it usually combines certain permission delegation mechanism to keep the flexibility as well as to fulfill PoLP. The sudo command in Unix-like OS is an example that allows users to be temporarily assigned to privileged roles that have minimal permissions to complete certain tasks. Leading cloud providers such as AWS also provide RBAC features that enable system administrators to restrict who can have access to which part of cloud resource/services by assigning proper roles to users.

However, it is difficult to avoid identities to be over-permissioned when multiple systems have close collaboration to each other. It frequently happens in enterprise that an internal core system provides more and more services, so other collaborating systems would have to request additional roles to acquire more permissions in order to consume the new services. The core system would be at higher security risk with more roles that are aggregately assigned to the same collaborating systems. Leave aside the malicious security breach, erroneous code from other systems may unintentionally exploit additional permissions and accidentally modify/remove data on the core system.

DESCRIPTION

This document claims a method to add a proxy module that can be dynamically switched to different granted roles in order to decouple different business functional requests and to be executed under the least permissions. So even if a system was assigned a superset of roles that are required by all of the possible actions this system is allowed to do, each action will be executed as a proper role with minimal permission to avoid system breach. Fig. 1 shows the overview of the flow (three actions as an example).

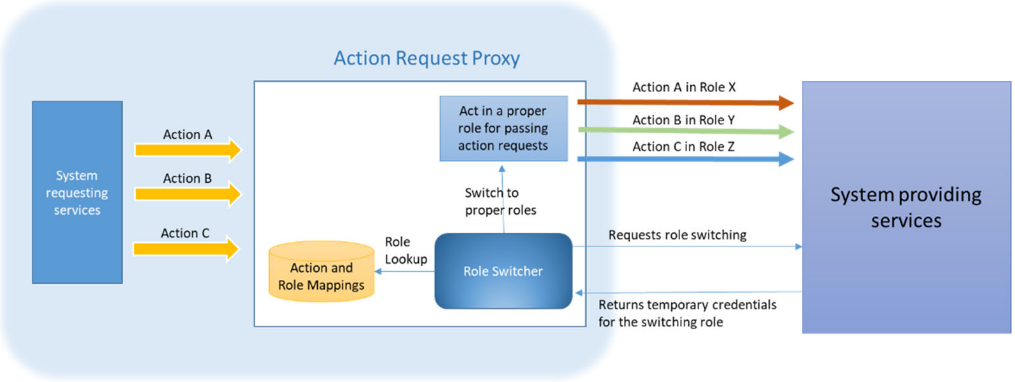


Figure 1 the overview of the flow

Fig. 2 illustrates the module is in the role with only basic permission when no action is requested. The basic permission cannot do anything but delegate permission to itself. So even if the module is compromised, the hacker cannot use its credential to access other systems.

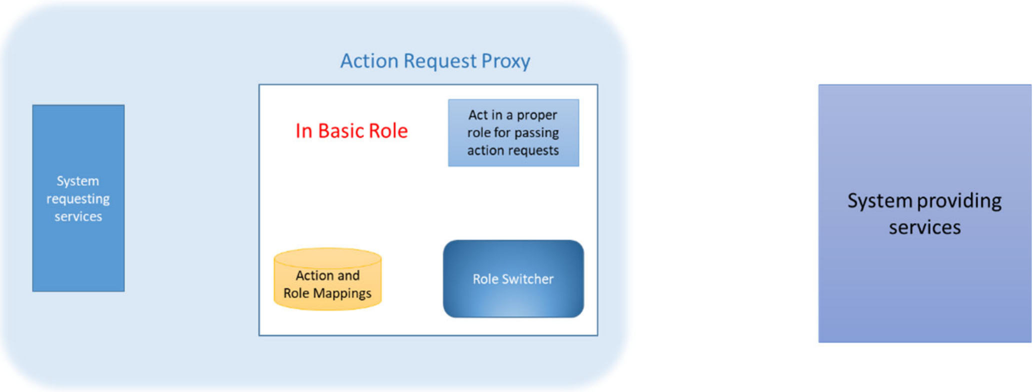

Action Request Proxy
In Basic Role
Act in a proper role for passing action requests
System requesting services
Action and Role Mappings
Role Switcher
System providing services

Figure 2

Fig. 3 describes the following two parts:

1. All of the actions from the system on the permission grantee side will send the action requests to the system on the permission granter side through the proxy module.
2. The module knows how to map the roles and actions so the minimal permissions will be granted for completing the corresponding actions.

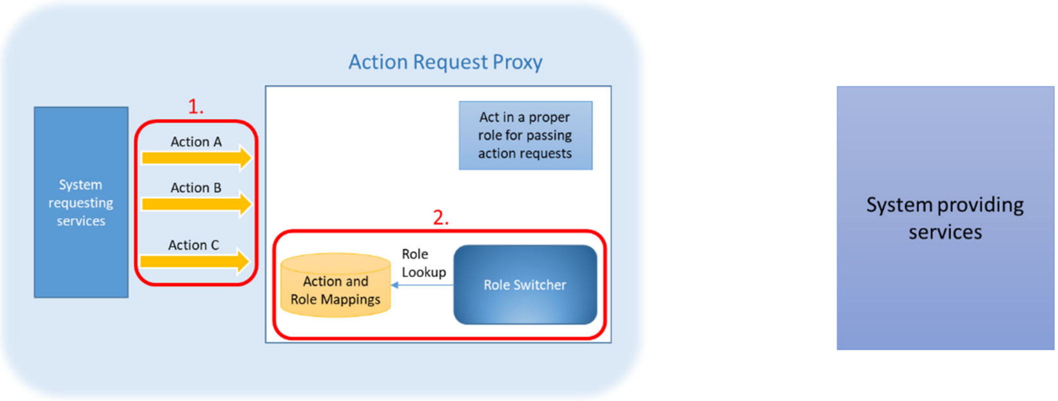

Action Request Proxy
1.
Action A
Action B
Action C
System requesting services
Act in a proper role for passing action requests
2.
Role Lookup
Action and Role Mappings
Role Switcher
System providing services

Figure 3

Action A: assume Action A requires the permissions granted to Role X (see Fig. 4):

1. The proxy module requests temporary credentials for the proper role it is about to switch to (Role X in the example). The system providing services will return the temporary credentials of Role X to the module.
2. The module uses the returned credentials to switch to Role X.
3. The module plays Role X and passes Action A to the system providing services.

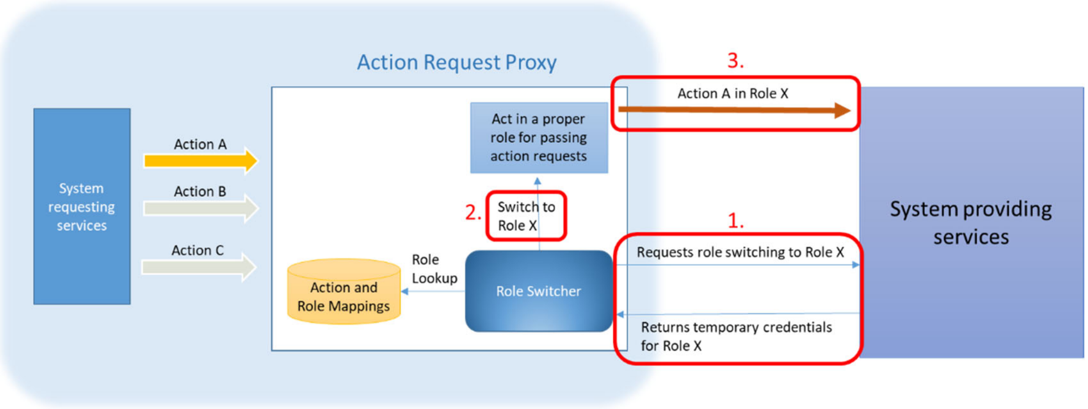

Action Request Proxy
System requesting services
Action A
Action B
Action C
Act in a proper role for passing action requests
2.
Switch to Role X
Role Lookup
Action and Role Mappings
Role Switcher
3.
Action A in Role X
1.
Requests role switching to Role X
Returns temporary credentials for Role X
System providing services

Figure 4

Action B: assume Action B requires the permissions granted to Role Y (see Fig. 5):

1. The module request credential for switching to Role Y
2. The module uses the returned credentials to switch to Role Y.
3. The module plays Role Y and passes Action B to the system providing services.

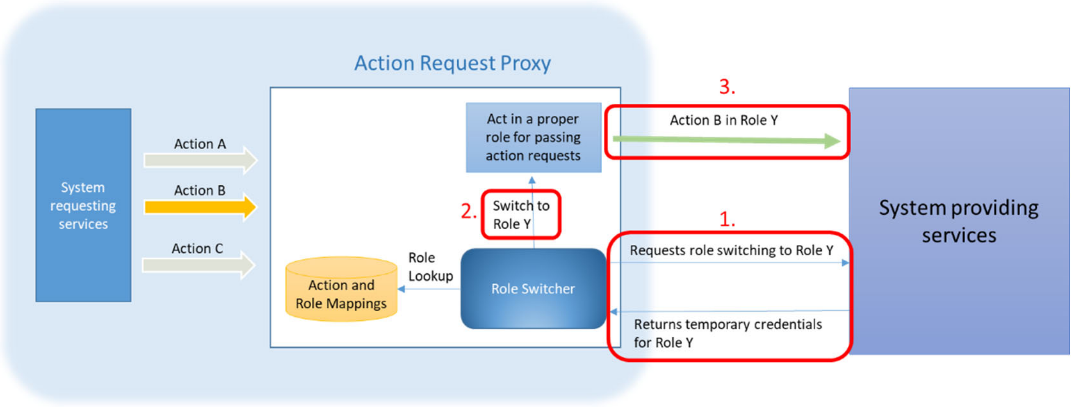
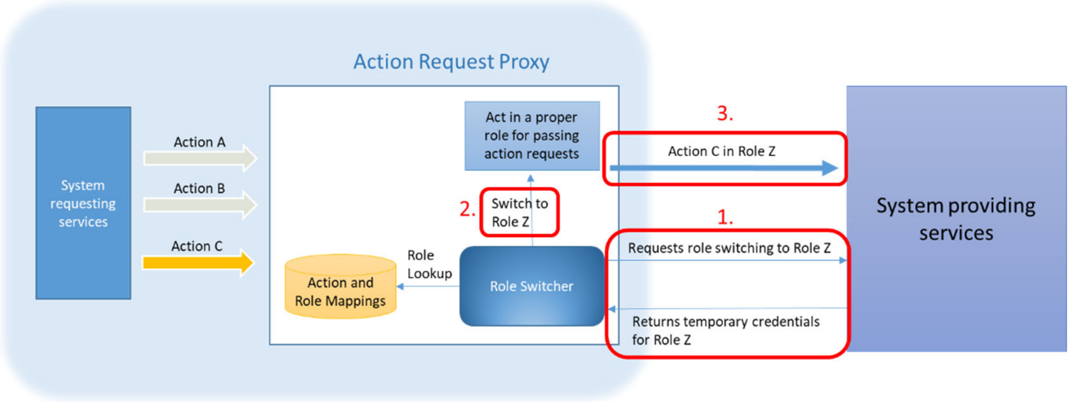

Action Request Proxy
System requesting services
Action A
Action B
Action C
Act in a proper role for passing action requests
2.
Switch to Role Y
Role Lookup
Action and Role Mappings
Role Switcher
3.
Action B in Role Y
1.
Requests role switching to Role Y
Returns temporary credentials for Role Y
System providing services

Figure 5

Action C: Assume Action C requires the permissions granted to Role Z (see Fig. 6):

Run the same steps as the other actions to complete action C in Role Z.


Action Request Proxy
System requesting services
Action A
Action B
Action C
Act in a proper role for passing action requests
2.
Switch to Role Z
Role Lookup
Action and Role Mappings
Role Switcher
3.
Action C in Role Z
1.
Requests role switching to Role Z
Returns temporary credentials for Role Z
System providing services

ASDFASDF

CONCLUSION

In this disclosure, we are introducing a new security mechanism to enforce principle of least privilege. This disclosure claims a method to consider adding a proxy module that can be dynamically switched to different granted roles in order to decouple different business functional requests and to be executed under the least permissions. This disclosure helps the granularity of access control reach to the level of request/transaction to prevent security breach from using over permissioned roles.

Disclosed by Hong-Wei Chou, William Tai, Winston Lee, Jason Chen, HP Inc.