A Secured Access Control for Software defined Networks using Reactive Programming

Aishvarya S SuryaBala B

Coimbatore Institute of Technology Coimbatore Institute of Technolgy

Affiliated to Anna university, Chennai, India Affiliated to Anna university,Chennai, India

ABSTRACT:

The objective of the paper is to propose a secured access control technique for software defined networks using reactive programming . The quality of services and access control of multiple networks can be controlled and improvised software defined mechanism or controller when compared with the dedicated hardware .a different types and level of security mechanisms are to be provided based on the networking policy and the clients requests for on demand applications a mobile network , home device network and an enterprise network with a lot of main frame servers to be provided with authentication and authorization based on the events occurred outside the client work space .a functional reactive programming is the solution to address the multiple network access and also the provision of expected quality of service based on the available bandwidth .the network, interface standards are used with physical and virtual networks so that the access control for the each information packets is achieved by a individual control flow technique.

INTRODUCTION:

Today’s tremendous increase of online applications usage and cloud services by the mobile connected and other devices results increase in carrier network traffic . In computer networking, network traffic control which is the process of managing,controlling or reducing the network traffic, particularly internet bandwidth where needs to ensure availability of network resource, security ,quality of service ,all without increasing operating and equipment costs. SDN (software defined network) has emerged as the industry’s response to meet these challenges because it is valuable but uses extend to other parts of network for increased network flexibility ,dynamic traffic flow, decreased latency and QoS.SDN is the physical separation of the network control plane (controller) from the forwarding plane, and where a control plane controls several device. It is an approach to computer networking that allows network administrators to manage network services through abstraction of higher-level functionality. With the help of SDN, network architecture can be instantly adjusted and that highly respond to applications and user requests. Services can be introduced far quickly, easily and at a lower cost. This makes it possible to optimize networks on the fly and quickly respond to changes in network usage without the need for manually reconfiguring existing infrastructure or purchasing new hardware. It seperates the control of network devices from the data they transport, and the switching software from the actual network hardware. It allows them to implement multiple logical network topologies on a single common network fabric. It has a comprehensive view of the entire network and its status, and with which switches (network resources) and application (network consumers) can communicate in real-time. Open flow is a communication protocol that gives access to the forwarding plane of a network switch or router over the network. It allows researchers to experiment their abstraction in heterogeneous environment in a uniform way at line-rate and with high port density; while on the other hand, manufacturers need not reveal the interior design and operations of their switches. IP networks are unable to match these qualities and because of this an increasing amount of Internet providers and business are beginning to rely more on SDNs.

ISSUES IN NETWORK ACCESS CONTROL:

The user's authentication method is not allowed:

This occurs because of problems with the server-side connection request policy when client systems use or virtual private network (VPN) enforcement. The connection request policy must be configured to override the network policy authentication settings. If the network policy authentication settings are not overridden, the company's (NPS) will reject the NAP client access requests using 802.1X and VPN connections. Access the NPS and configure the connection request policy for 802.1X and VPN connections to override the network policy authentication.

The client is incorrectly classified as non-NAP:

A client computer should support Network Access Protection, but the server-side says the client is non-NAP and refuses to grant full network access. Instead, the client is forced to adhere to a non-NAP policy (if you have one). This usually happens because of a snafu on the client end; usually the NAP Agent service, the NAP enforcement client or health checks are not enabled on the client.

ACCESS CONTROL TO SOFTWARE DEFINED NETWORK:

Using strong username and passwords lowers overall risk of a security breach, they strong effective security controls in a software defined network. When a client/server connection is authenticated:

* The application on the client side of the connection sends credentials to the server using the SSPI function InitializeSecurityContext (General).
* The application on the server side of the connection responds with the SSPI function AcceptSecurityContext (General).
* The SSPI functions InitializeSecurityContext (General) and AcceptSecurityContext (General) repeat until all the necessary authentication messages have been exchanged to either succeed or fail authentication.
* After the connection has been authenticated, the LSA on the server uses information from the client to build the security context, which contains an access token.
* The server can then call the SSPI function ImpersonateSecurityContext to attach the access token to an impersonation thread for the service.
* Stored User Names and Passwords contains invalid or incorrect credentials for a specific resource, access to the resource will be denied and the **Stored User Names and Passwords** dialog box will not appear.



IMPLEMENTATION:

Validation of username and password in network access control using (ELM) reactive programming

Sample code:

view model =

   form [ id "signup-form" ] [

       h1 [] [ text "Sensational Signup Form" ],

       label [ for "username-field" ] [ text "username: " ],

       input [ id "username-field", type' "text", value model.username ] [],

       label [ for "password"] [text "password: " ],

       input [ id "password-field", type' "password", value model.password ] [],

       div [ class "signup-button" ] [ text "Sign Up!" ]

   ]

Functional view model:

function view(model) {

   return form([ id("signup-form") ], [

       h1([], [ text("Sensational Signup Form") ]),

       label([ for("username-field") ], [ text("username: ") ]),

       input([ id("username-field", type\_("text"), value(model.username) ], []),

       label([ for("password") ], [ text("password: ") ]),

       input([ id("password-field", type\_("password"), value(model.password) ], []),

       div([ class("signup-button") ], [ text("Sign Up!") ])

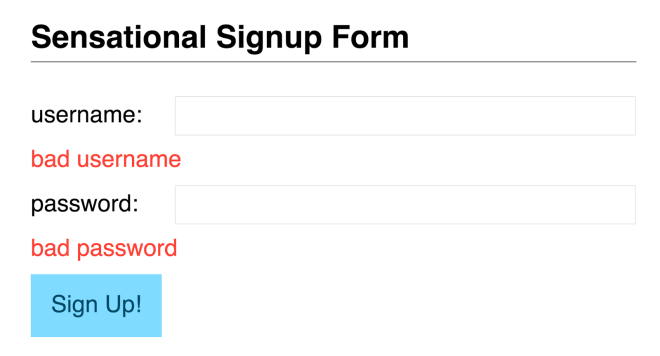
   ]);

}

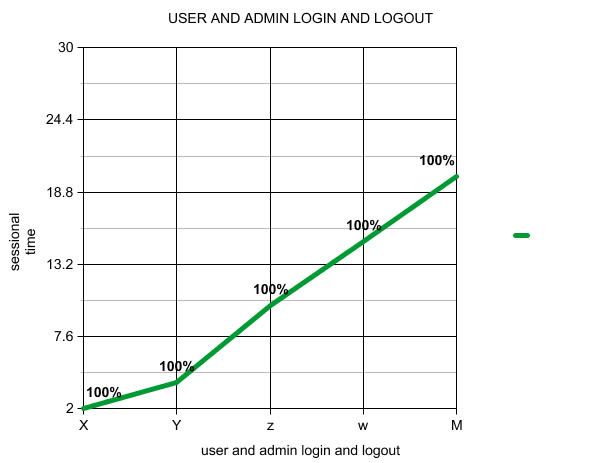
|  |
| --- |
| initialErrors = |
|  |

|  |
| --- |
|  |
| { username = "", password = "" } |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
| getErrors model = |
|  |
|  |
| { username = |
|  |
|  |
| if model.username == "" then |
|  |
|  |
| "Please enter a username!" |
|  |
|  |
| Else |
|  |
|  |
| "" |
|  |
|  |
|  |
|  |
|  |
| password = |
|  |
|  |
| if model.password == "" then |
|  |
|  |
| "Please enter a password!" |
|  |
|  |
| Else |
|  |
|  |
| "" |
|  |
|  |
| }  **Asking a admin about username availability**  var myInput = document.getElementById( "myMetaBoxInput" );  //Get form from input  var form = myInput.form;  //They have something already set  if ( form.onsubmit )  {  var previousSubmit = form.onsubmit;  form.onsubmit = function() {  //do my code....  //Call their function  previousSubmit();  }  } else form.onsubmit = myfunction;  **Asking a server about username availability**  if action.actionType == "VALIDATE" then     let         url =             "https://api.github.com/users/" ++ model.username         usernameTakenAction =             { actionType = "USERNAME\_TAKEN", payload = "" }         usernameAvailableAction =             { actionType = "USERNAME\_AVAILABLE", payload = "" }         request =             Http.get (succeed usernameTakenAction) url         neverFailingRequest =             Task.onError request (\err -> Task.succeed usernameAvailableAction)     In         ({ model | errors <- getErrors model }, Effects.task neverFailingRequest) |
|  |
|  |
|  |
|  |

|  |
| --- |
|  |



A GRAPH SHOWING THAT CLIENTS AND ADMIN IN A NETWORK LOGIN AND LOGOUT:



REACTIVE [PROGRAMMING FOR ACCESS CONTROL:

Checking whether the given username is available. In Elm represent Actions as data, it represents effects as data too specifically, using values called Tasks. Tasks have things in common with both Promises and callbacks from JavaScript The ELM effects library lets run Tasks in a way that always results in an Action being fed back into update function, which can neatly incorporate effects into the architecture that have been following this entire time.

The following are among the new security-related features in reactive programming for access control:

* Software Restriction policies.

New security policy options allow you to prevent certain software applications from running based on a file path, Internet zone, certificate, or hashed file path.

* Fast user switching.

Computers that are not connected to a domain, users can switch from one user account to another without logging off or closing their applications.

* Stored user names and passwords.

This utility provides secure storage for user names and credentials needed to access network or Internet resources.

* New service accounts.

This includes two new service accounts, LocalService and NetworkService, to enable graduated levels of permissions on services. Services can run as LocalService on the local computer, as NetworkService on the network, or as part of the Local System. Any service not running as one of the three built-in service accounts must have its own account.

* Password Reset Wizard.

This wizard makes it possible for users to create a secure reset disk, which they can use at a later date in case they forget the password for their local account.

This changes are in four basic areas:

* Improved network protection
* Improved memory protection
* Improved e-mail security
* Improved browser security

Validation

* Manages local security policy
* Manages audit policy and settings
* Generates tokens that contain user and group information as well as information about the security permissions for the user

REFERENCES

[1] chandrasekran subramaniam, vikneshkumar T, vignesh R, “conflict management model using functional reactive programming in software defined networking“, Proceedings of 6th SARC-IRF International Conference, 06th July-2014

[2] Yosr Jarraya, *Member, IEEE*, Taous Madi, and Mourad Debbabi, *Member, IEEE, “*A Survey and a Layered Taxonomy ofSoftware-Defined Networking”

[3] Mandar B shinde , sunil G. Tamhankar , “ Review: software defined networking and open flow “, International journal of scientific research in network security and communication, vol -1,issue -2,ISN 2321-3256 .

[4] Manal algarni, vinayak nair,david martin,sayali shirgaonkar,”software defined networking overview and implementation”