WEBSITES SECURITY POLICIES IMPLEMENTATION USING ALLOY ANALYZER

A PREPRINT

Renzo Carpio[*]
Engineering and Computer Science
Syracuse, New York
rcarpio@syr.edu

Coauthor
Izzat Alsmadi Affiliation
Texas A&M University, San Antonio Address
San Antonio, TX 78224 ialsmadi@tamusa.edu

October 10, 2021

ABSTRACT

This paper focuses in Policies for Websites. Through this paper a website security policy will try to explain the main components that are present in a website related to security. The approach introduces all the elements that are present in a website at the software and the hardware level. Additionally and to reflect website dynamic behaviors, we will present the elements that are present during a website session and interactions between a user and the website.

Keywords Alloy · Analyzer · Security · Policies · WebSite · Assurance Foundations · Syracuse

1 Introduction

For my project in this class the purpose is to showcase website security policies using Alloy Analyzer for modeling these scenarios, there are many situations in which there is the need for companies to ensure that their operations are hundred percent reliable, nowadays most organization around the world depends on reliable web interactions between the different parties involved. It is crucial for an organization to ensure that a web site request to and from its premises is always reliable meaning it can ensure integrity, authenticity, reliability, and availability. Almost all companies around the world have a website which could be just menial information, or it could be extremely important information pertaining to the company. That is why is so important to have website security policies in place that ensure that these communications meet these requirements. Alloy Analyzer will help to model and test these critical scenarios that are present in a website.

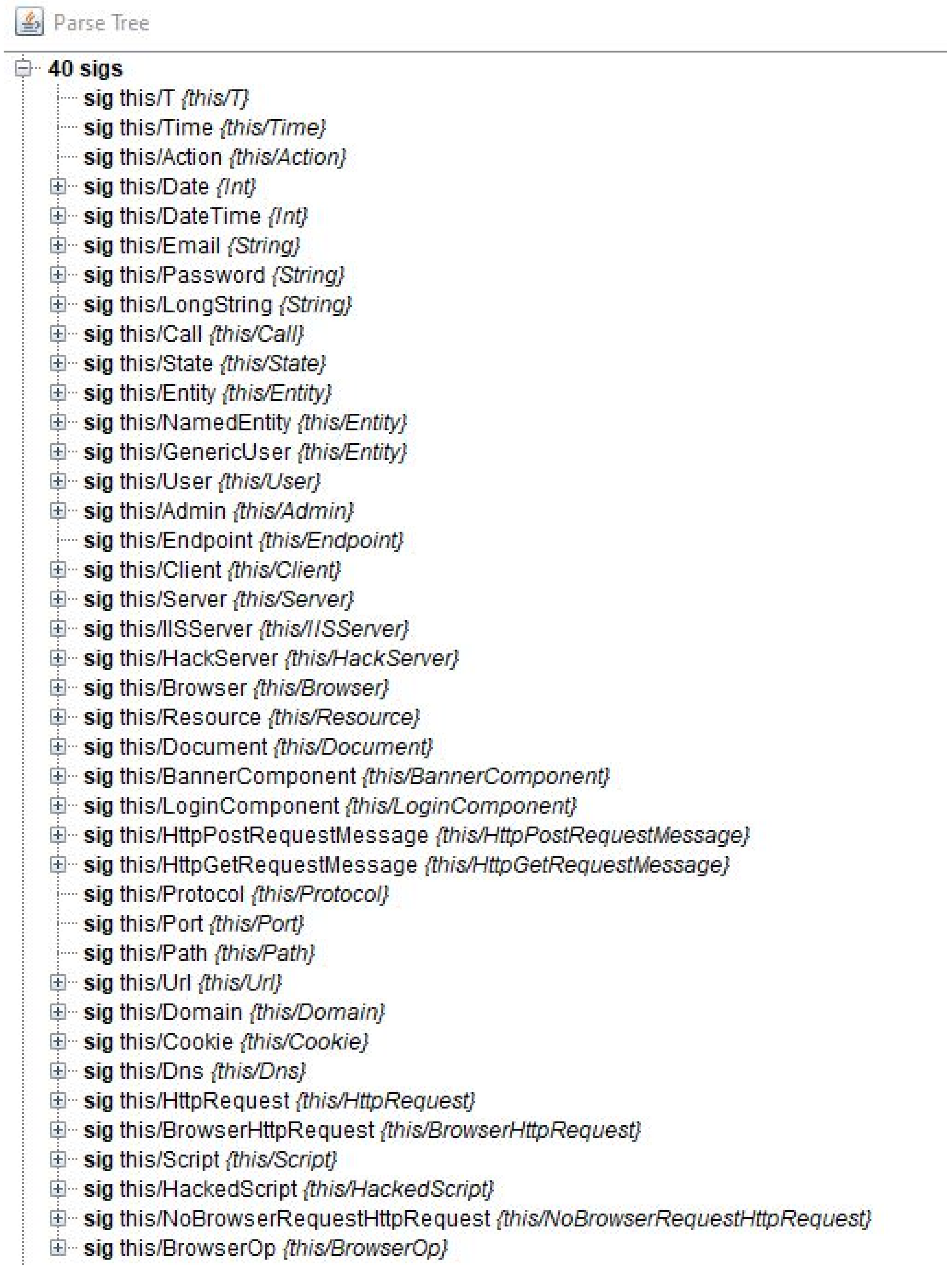
There is a section in this final report were papers similar to my project are analyzed one of the model that this project tries to showcase is the that there is a known attack which is called cross site forgery, in this type of attacks a end user is forced or unseemly unaware to execute malicious code in the background, the user could be either authenticated or not.. This is an advances model but we have tried to show a basic scenario of this specific situation. In my case and given the main topic of my project, "Website Security Policies" some of the code examples that are described in sections below, are how the whole web request from the endpoint, client browser in my case, all the way to the response, in my case access to resources. Different constraints are described which are applied in this project: enforce an origin, require a response, request direction, and some other which can observed on the code.

[*]Renzo Carpio - Alloy Analyzer - Security Policies WebSite.)—

2 Web Policies Model

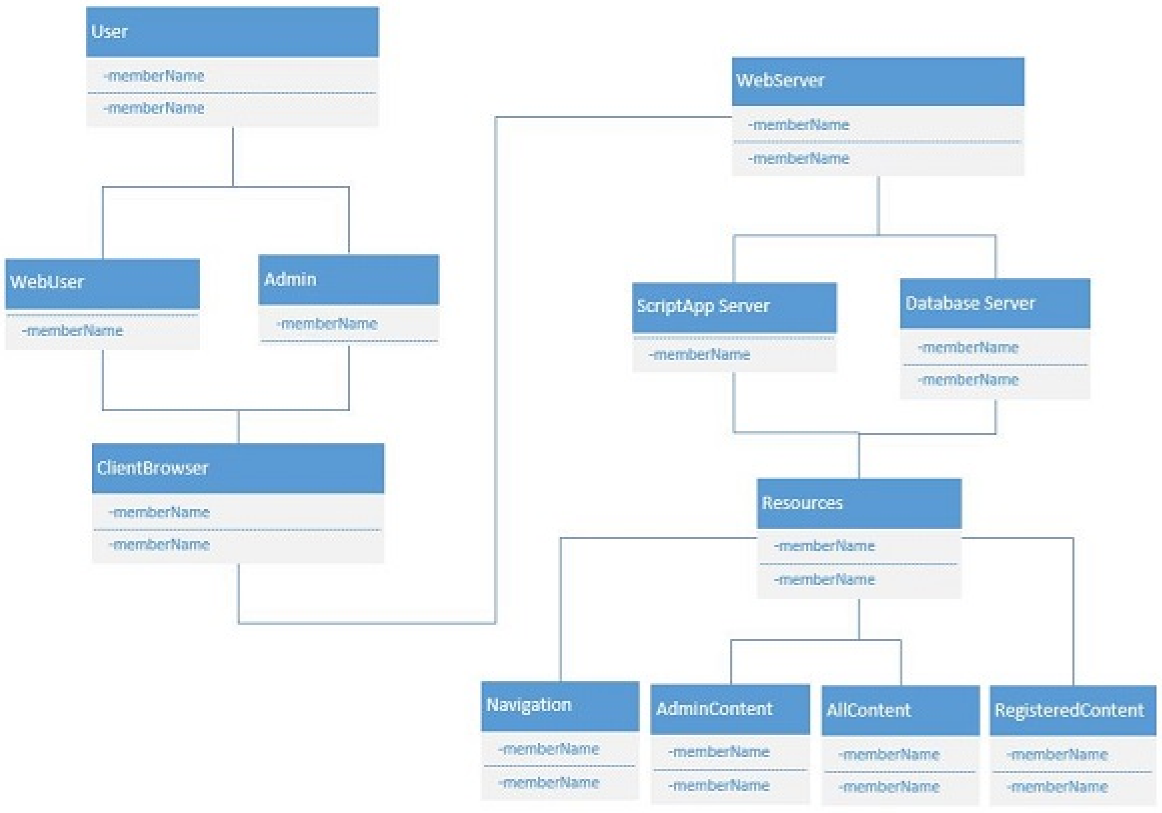
2.1 Model Entities

The Following entities are present in my model, Security Policies WebSite. The state of the model is determined by the relationships among objects and the membership of objects in sets these can change in time.



```
40 sigs
  sig this/T {this/T}
  sig this/Time {this/Time}
  sig this/Action {this/Action}
  sig this/Date {Int}
  sig this/DateTime {Int}
  sig this/Email {String}
  sig this/Password {String}
  sig this/LongString {String}
  sig this/Call {this/Call}
  sig this/State {this/State}
  sig this/Entity {this/Entity}
  sig this/NamedEntity {this/Entity}
  sig this/GenericUser {this/Entity}
  sig this/User {this/User}
  sig this/Admin {this/Admin}
  sig this/Endpoint {this/Endpoint}
  sig this/Client {this/Client}
  sig this/Server {this/Server}
  sig this/IISServer {this/IISServer}
  sig this/HackServer {this/HackServer}
  sig this/Browser {this/Browser}
  sig this/Resource {this/Resource}
  sig this/Document {this/Document}
  sig this/BannerComponent {this/BannerComponent}
  sig this/LoginComponent {this/LoginComponent}
  sig this/HttpPostRequestMessage {this/HttpPostRequestMessage}
  sig this/HttpGetRequestMessage {this/HttpGetRequestMessage}
  sig this/Protocol {this/Protocol}
  sig this/Port {this/Port}
  sig this/Path {this/Path}
  sig this/Url {this/Url}
  sig this/Domain {this/Domain}
  sig this/Cookie {this/Cookie}
  sig this/Dns {this/Dns}
  sig this/HttpRequest {this/HttpRequest}
  sig this/BrowserHttpRequest {this/BrowserHttpRequest}
  sig this/Script {this/Script}
  sig this/HackedScript {this/HackedScript}
  sig this/NoBrowserRequestHttpRequest {this/NoBrowserRequestHttpRequest}
  sig this/BrowserOp {this/BrowserOp}
```

2.2 Model Entities Schema



3 Alloy Analyzer - Code Policies for Websites

Listing 1: Alloy Code

```

//=========================================
//==== Syracuse University
//==== Assurance Foundations
//==== Alloy Model - Security Website Policies
//==== Renzo Carpio
//=====================================

module RenzoCarpioWebAlloySecurityPolicies[TParam]

open util/boolean
open util/relation as rel
open util/ordering[Time] as ord
open util/ordering[State]

sig Time {}
sig Action {}
sig Date = Int {}
sig DateTime = Int {}
sig Email = String {}
sig Password = String {}
sig LongString = String { }

abstract sig RequestCall { start, end: Time, from, to: TParam }

//Sig State Definition
sig State{
  counter: Int,
  startState: Int,
  nextState: Int,
  prevState: Int,
  endState: Int,
  nxt: lone State
}

//Generic Entity from which other Entities will extenend shared properties
abstract sig Entity {
   name:     lone String,
   firstname:     lone String,
   lastname:     lone String,
   email:     lone String,
   owners:   set Entity,
   created:  DateTime,
   modified: DateTime,
}

//Extends from generic Entity / Users will extend from this entity to highlight
     the use of a common attribute as name
abstract sig NamedEntity extends Entity { } {
  some name
}

//This Entity in the model represents a Generic User in the Website
abstract sig GenericUser extends NamedEntity {
  suspended: Bool,
  active: Bool,
  visitor: Bool,
  registeredEmail:     disjoint Email,
  password:  Password,
  dob: DateTime
}

//This Entity extends from the "GenericUser" in the model represents a Regular
    User in the Website
sig User extends GenericUser {
  userprofilename : lone LongString,
  bio : lone LongString,
  registered : Bool,
  confirmedEmail : Bool,
  getProfile : Bool
  //groups = regulars.this
}

//This Entity extends from the "GenericUser" in the model represents a
    Administrator in the Website
sig Admin extends GenericUser {
  userprofilename : lone LongString,
  bio : lone LongString,
  admin : Bool,
  confirmedEmail : Bool,
  registered : Bool,
  getProfile : Bool
}

//A Website interaction has EndPoints
/*
-User enters a Website URL in any client (Phone, Table, Computer, etc.) this
    client uses a browser which could be Edge, Chrome, Firefox, Safari, etc.
-One that a URL is entered the browse sends a request to the specific web
    server that is trying to reach
-One that the server is reached then the server returns a response as a HTML
    page or if the request is not a regular HTTP request, then the result could
    be in any other format (xml, json, etc.)
-Once that the response is received then the browse renders the result in the
    user device
*/
abstract sig Endpoint {}

/*This entity represents a Client which is an EndPoint starting a request - In
    this model a client starting this requests  can be a User or an Admin */
abstract sig Client extends Endpoint {
  browserusersession: set GenericUser,
  browserClient : lone LongString
}

/*This entity represents a Server which is also part of and EndPoint a request
    */
abstract sig Server extends Endpoint {
   nameServer:     lone String,
   typeServer:     lone String,
    resources: Path -> lone Resource
}

//Different type of Servers inheriting from Main entity Server
one sig IISServer, HackServer extends Server  {
  ipAddress:     lone String
}

//Client starts a request using a Browser
sig Browser extends Client {
  clientDocuments: Document -> Time,
  clientCookies: Cookie -> Time
}

```

```
//This entity represent a Generic entity for resource that need to be accessed
    in an interaction
sig Resource {
   nameResource:     lone String,
   typeResource:     lone String,
  parent: lone Resource,
  date : Date,
  name : lone LongString,
  descr : lone LongString
}

//This  entity represents artifact that is present in the request
sig Document {
  src: Url,
  nameDocument:     lone String,
  typeDocument:     lone String,
  content: Resource -> Time,
  domain: Domain -> Time
}

//When a request is created for a document / A example will be requesting a
    specific Banner for the Website
sig BannerComponent extends Document {

}

//When a request is created for a document / A example will be requesting a
    login component for the Website
sig LoginComponent extends Document {

}

//Entity that simulates a Post Request
sig HttpPostRequestMessage extends BrowserOp {
  messageType: Resource,
  targetInitialOrigin: HttpPostRequestMessage,
  targetInitialDestination: HttpPostRequestMessage
}

//Entity that simulates a Get Request
sig HttpGetRequestMessage extends BrowserOp {
  messageType: Resource,
  targetInitialOrigin: HttpGetRequestMessage,
  targetInitialDestination: HttpGetRequestMessage
}



//This entity represents the key components that need to be present in a
    request
/*
Protocol: http, https, ftp
Host: What domain is this resource hosted www.renzocarpio.com
Port: Port of communication http 80 / https 443
Path: This is the PATH of the request it could be a GET/ POST / UPDATE / DELETE
Ex. /Content/CIS634.html which is a request to get the following resource /
    Content/CIS634.html from the server.
*/
sig Protocol, Port, Path {}
sig Url {
  finalHost: Domain,
  finalPort: lone Port,
  selectedProtocol: Protocol,
  selectedPath: Path
}
```

```

//This entity represents a particular website domain /
sig Domain {
  subsumes: set Domain
}

//This entity represent Cookies present in a Site Request
sig Cookie {
  domains: set Domain
}

//Domain Name Server
/*
A DNS main purpose is to find a domain name, it will search for that domain
    using the
IP address that was associated with the request, one that the IP is identified
    then the
domain is found and a connection is established which will allow a users
    browser and
a server to connect and request and transmit information.
*/
one sig Dns {
  map: Domain -> Server
}


//An entity that represent a HTTP Request
abstract sig HttpRequest extends RequestCall {
  url: Url,
  urlOriginin: Url,
  urlDestination: Url,
  body: lone Resource,
  response: lone Resource,
  sendSetCookies: set Cookie -> Time,
  receiveStoredCookies: set Cookie -> Time,

}{
  //from in Client
  //to in Dns.map[url.host]
}


//A type of HTTP REQUEST generated from client browser
sig BrowserHttpRequest extends HttpRequest {
  document: Document

}{
  --  in This section the document contains wht to render when a request is
      generated
  clientDocuments.end = clientDocuments.start + from -> document
  -- when a response is formed it contains all the content
  content.end = content.start ++ document -> response
  -- this particualr section section indicates document which has url domain of
       its host
  domain.end = domain.start ++ document -> url.finalHost
  -- this particualr section indicates doc originating field of the request
      from the User
  document.src = url
}

//Entity that represents "Client-Site scripts" (Ex. Javascript files)
sig Script extends Client { context: Document }

//Different tye of script to access Documents and Resources
sig HackedScript extends Script {}
```

```

//A type of HTTP REQUEST not generated from client browser / Ex (Program
    Requesting Information)
sig NoBrowserRequestHttpRequest extends HttpRequest {}{
  noClientBrowserChange[start, end] and noDocumentResourcesChange[start, end]
}


abstract sig BrowserOp extends RequestCall { doc: Document }{
  --noBrowserChange[start, end]
}

/* ASSERTIONS*/
check {
    all resource1, resource2: HttpRequest | resource1.url = resource2.url
        implies resource1.response = resource2.response
} for 3

fact ServerAssumption {
  all server1, server2: Server |
    (some Dns.map.server1 & Dns.map.server2) implies server1.resources =
        server2.resources
}

fact singleUserPerBrowserSession{
  //For each Uder 'u'
  all u: GenericUser |
    //there is exactly one Account 'a' per request
      one a: Client |
        //for which 'u' belongs to 'a' that specific client request
        u in a.browserusersession
}

fact everyUserMustHaveUniqueName {
  all disj a, b: GenericUser | a.name != b.name
}

fact  noResourcesCycles{
  //There is not Resource 'r' for which
  no r: Resource |
    // 'r'  is in its own parent chain
    r in r.^parent
}


fact noSelfLinks {
  all x: HttpRequest | x.urlOriginin != x.urlDestination
}
/*
no links from a host to itself
fact noSelfLinks {all x: Link | x.from != x.to}
same as above
*/

/*PRE-CONDTIONS*/
//Check that not Document have Changed time is implented to check integrity of
    request, no attack has been performed

//Assertions
assert  NoSharedBrowserSession{
  //For every Resource 'r'
  all r: GenericUser |
    //There is exactly one Account 'a' for which
    all a:Client |
      // 'r' belongs tp 'a'
      r in a.browserusersession
}


pred noClientBrowserChange[startTime, endTime: Time] {
  clientDocuments.endTime = clientDocuments.startTime and clientCookies.endTime
       = clientCookies.startTime
}
pred noDocumentResourcesChange[startTime, endTime: Time] {
  content.endTime = content.startTime and domain.endTime = domain.startTime
}

run noClientBrowserChange
run noDocumentResourcesChange

//Check that there is at least one domain server available to establish
    communication
pred atLeastOneDomainServer{
  one Dns
}

//At least one User during this Interaction
pred atLeastOneUserRequest{
  one User && one HttpRequest
}




/*POST-CONDTIONS*/
pred inv [u: User] {}

pred invAdmin [a: Admin] {}

pred invClientCurrentSessionn[c:Client]{}

pred invServerCurrentSessionn[c:HttpRequest]{}

//=====================================
//==== CRUD OPERATIONS
//==== User Create/Read/Update/Delete
//=====================================
pred addUser [u, u': User] {
  u' = u
}

pred showUser [u, u': User] {
  inv[u]
  addUser[u, u']
}

pred removeUser [u, u': User] {
  u = u'
}

pred updateUser [u', uu': User] {
  u' = uu'
}

run addUser
run showUser
run removeUser
run updateUser
//=====================================


//=======================================
//==== CRUD OPERATIONS
//==== Admin Create/Read/Update/Delete
//=======================================
pred addAdmin [a, a': Admin] {
  a' = a
}

pred showAdmin [a, a': Admin] {
  invAdmin[a]
  addAdmin[a, a']
}

pred removeAdmin[a, a': Admin] {
  a = a'
}

pred updateAdmin [a', aa': Admin] {
  a' = aa'
}

run addAdmin
run showAdmin
run removeAdmin
run updateAdmin
//=======================================


//=======================================
//==== CRUD OPERATIONS
//==== Client Request Create/Read/Update/Delete
//=======================================
pred addClient [c, c': Client, bus: GenericUser ] {
  c'.browserusersession = c.browserusersession + bus
}

pred showClientCurrentSession [c, c': Client, bus: GenericUser] {
  invClientCurrentSessionn[c]
  addClient[c, c', bus]
}

pred removeEndClient[c, c': Client, bus: GenericUser ]  {
  c'.browserusersession = c.browserusersession - bus
}

pred updateClient[c', cc': Client, bus: GenericUser ]  {
    c'.browserusersession = cc'.browserusersession + bus
}

run addClient
run showClientCurrentSession
run removeEndClient
run updateClient
//=======================================


//=======================================
//==== CRUD OPERATIONS
//==== HttpReques Request Create/Read/Update/Delete
//=======================================
pred addHttpRequest [s, s': HttpRequest, res: Url] {
  s'.url = s.url + res
}

```

```
pred showHttpRequestCurrentSession [s, s': HttpRequest, res: Url] {
  invServerCurrentSessionn[s]
  addHttpRequest[s, s', res]
}

pred removeHttpRequestr[s, s': HttpRequest, res: Url ]  {
  s'.url = s.url - res
}

pred updateHttpRequest[s', ss': HttpRequest, res: Url ]  {
  s'.url = ss'.url + res
}

pred addUserHttpRequest [u,u':GenericUser, s, s': HttpRequest] {
  s'.url = s.url + u
}

run addHttpRequest
run showHttpRequestCurrentSession
run removeHttpRequestr
run updateHttpRequest
run addUserHttpRequest

//========================================



//========================================
//==== CRUD OPERATIONS
//==== Cookies Create/Read/Update/Delete
//========================================
pred addCookieHttpRequest [c, c': Cookie, s, s': HttpRequest, url: Url] {
  s'.sendSetCookies = s.sendSetCookies
}

pred deleteCookieHttpRequest [c, c': Cookie, s, s': HttpRequest, url: Url] {
  s.sendSetCookies = s'.sendSetCookies
}

pred updateCookieHttpRequest [c, c': Cookie, s', s'': HttpRequest, url: Url] {
  s'.sendSetCookies = s''.sendSetCookies
}


run addCookieHttpRequest
run deleteCookieHttpRequest
run updateCookieHttpRequest

//========================================

//========================================
//==== CRUD OPERATIONS
//==== DNS Create and Discard DNS RequestCall
//========================================
pred addDNSCall [dns, dns': Dns, ser: Server ] {
  dns'.map = dns.map
}

pred discardDNSCall [dns, dns': Dns, ser: Server ] {
  dns.map = dns'.map
}


run addDNSCall
run discardDNSCall
```

```


//========================================






//========================================


//========================================
//==== PARAMETRIC MODULE
//==== Oepn util/ordering
//========================================
//Declared at the top of code
// module RenzoCarpioWebAlloySecurityPolicies[T]

//Utility being used for this sample predicate
//open util/relation as rel

//Cookies are consumed in 2 ways either  User Cient Browser is setting
//a cookie fior the first time or if a cookie exists then that cookie is
//analyzed and updated accordingly this is an  acyclic operation

pred setUpRetrievecookie[c: Cookie -> Cookie]{
  rel/acyclic [c, Cookie]
}

run setUpRetrievecookie
//========================================



//========================================
//==== OPERATIONS ORDERING
//==== Open util/ordering
//========================================

pred init(t: Time){
  //All accounts have no resources at t=0
  //#User >  0
  //#Client.browserusersession > 0
}

//Fact that defines Adding and Removin a User Client Session
//Adding general traces

fact Trace{
  init[first] //initilize userstate
  all ug:GenericUser | all ht:HttpRequest |
    let ug' = ug.next | let ht'= ht.next |
      addUserHttpRequest[ug,ug', ht,ht' ]

}

fact AddingUserTraces{
  init[first]
  all u: User | let u' = u.next | {
    addUser[u,u'] or removeUser[u,u']
  }
}

//Fact that defines Adding and Removing a User Client Session
fact AddingUserClientTraces{
  init[first]
  all c: Client | let c' = c.next | {
    some gu:GenericUser {
      addClient[c,c',gu] or removeEndClient[c,c',gu]
    }
  }
}

run init

//=======================================


//=======================================
//==== OPERATIONS State
//==== open util/ordering[State]
//=======================================

//This PRED Initializes a State
pred InitialState [s:State]
{
  one s.startState
  no s.nextState
}

//This PRED Initializes a User Session
pred InitUserSession(s, s' : State, u: User){
  s'.nextState = s.startState + u
}

//This PRED Terminated a User Session
pred TerminateUserSession(s, s' : State, u: User){
  s.endState = s'.startState
}

//This PRED create a new HttpRequest where a HTTPRequest is involved, a
    resource and  User
pred NewHttpRequest [s, s': HttpRequest, res: Url, u: User] {
  s'.url = s.url + res + u
}

pred trans[t,t': Time]{
  (some u,u': GenericUser | addUser[u,u'] )
  or
  (some u,u': GenericUser | removeUser[u,u'] )
  or
  (some s, s': HttpRequest, res: Url  | addHttpRequest[s,s', res] )
  or
  (some s, s': HttpRequest, res: Url  | removeHttpRequestr[s,s', res] )
  or
  (some dns, dns': Dns, ser: Server | addDNSCall[dns,dns', ser] )
  or
  (some dns, dns': Dns, ser: Server  | discardDNSCall[dns,dns', ser] )
  or
  (some s, s': HttpRequest, cookie, cookie': Cookie, res: Url  |
      addCookieHttpRequest[cookie,cookie', s, s', res] )
  or
  (some s, s': HttpRequest,  cookie, cookie': Cookie, res: Url |
      deleteCookieHttpRequest[cookie,cookie', s, s', res] )
  or
  (some s, s': HttpRequest, cookie, cookie': Cookie, res: Url |
      updateCookieHttpRequest[cookie,cookie', s, s', res] )

}

```

```


run InitialState
run InitUserSession
run TerminateUserSession
run NewHttpRequest
run trans




//=======================================
//==== OPERATIONS TIME
//==== Open util/ordering
//=======================================



pred createHttpPostRequestMessage [s, s': HttpPostRequestMessage, res: Url] {
  s'.messageType = s.messageType + res
  s'.targetInitialOrigin = s.targetInitialOrigin
  s'.targetInitialDestination = s.targetInitialDestination
}

pred createHttpGetRequestMessage [s, s': HttpGetRequestMessage, res: Url] {
  s'.messageType = s.messageType + res
  s'.targetInitialOrigin = s.targetInitialOrigin
  s'.targetInitialDestination = s.targetInitialDestination

}

run createHttpPostRequestMessage
run createHttpGetRequestMessage



//=======================================

//=======================================
//==== Time
//==== open util/ordering[State]
//=======================================
//Select 5 relations/operations in your project and make sure to inject Time
    entity
// Relations show as comments below but implemented in the code above

// 1)  documents: Document -> Time,
// 2)  cookies: Cookie -> Time
// 3)  content: Resource -> Time,
// 4)  domain: Domain -> Time
// 5)  sendSetCookies: set Cookie -> Time,
// 6)  receiveStoredCookies: set Cookie -> Time,



//=======================================



/*
pred example { }
run example for exactly 5 HttpRequest, 5 User
*/

//=======================================
//==== Execution
//==== Generate 4 instances for every entity in the
//==== WebSite Security Policies Model
//=====================================
run {} for 4
```

4 Website Security Policies

4.1 Policy 1 - Session cannot be shared - Single User per Browser Session

For this example policy there is a check to prevent that two users share the same session, in this policy what the model is trying to emulate is how to prevent a men in the middle attack. This policy is a policy that forces that a single user has access to to a single browser session.

Listing 2: Alloy Code

```

//Assertions
  assert  NoSharedBrowserSession{
    //For every Resource 'r'
    all r: GenericUser |
      //There is exactly one Account 'a' for which
      all a:Client |
        // 'r' belongs tp 'a'
        r in a.browserusersession
}
```

4.2 Policy 2 - Check Valid URL from Requests

For this example policy there is a check to see if a request is valid or not. Dns is involved to check if the URL exisits and if the URL exists then the resource is available. The server will also send a response, which is a code from the server that the Browser will received to see if the request was valid and successful.

Listing 3: Alloy Code

```

/* ASSERTIONS*/
check {
    all resource1, resource2: HttpRequest | resource1.url = resource2.url
        implies resource1.response = resource2.response
} for 3

fact ServerAssumption {
  all server1, server2: Server |
    (some Dns.map.server1 & Dns.map.server2) implies server1.resources =
        server2.resources
}
```

5 Website Security CRUD Operations

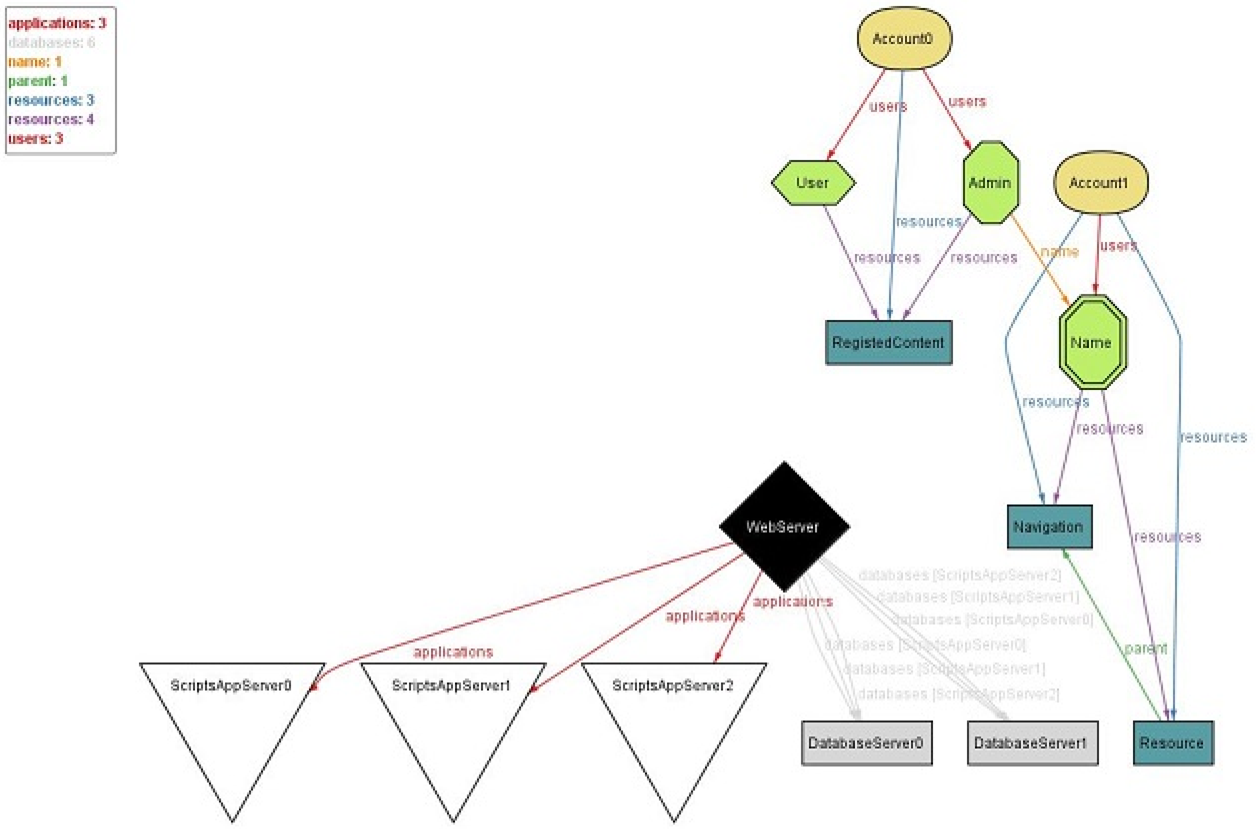
5.1 CRUD User Entity

Explanation
CRUD User Entity - Use Case - Users - Create, Read, Update and Delete

Listing 4: Alloy Code

```

//========================================
//==== CRUD OPERATIONS
//==== User Create/Read/Update/Delete
//========================================
pred addUser [u, u': User] {
 u' = u
}

pred showUser [u, u': User] {
 inv[u]
 addUser[u, u']
}

pred removeUser [u, u': User] {
 u = u'
}

pred updateUser [u', uu': User] {
 u' = uu'
}

run addUser
run showUser
run removeUser
run updateUser
//========================================
```

screenshot



5.2 CRUD Admin Entity

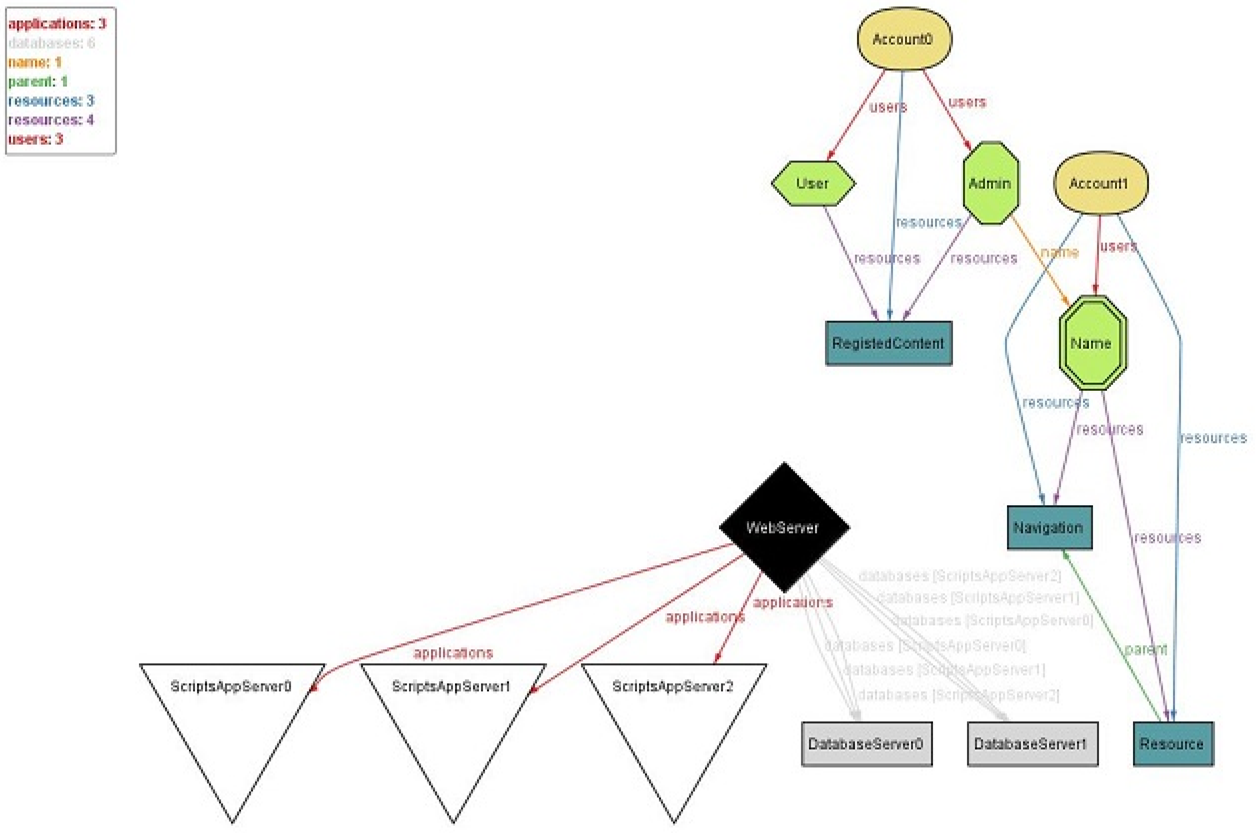
Explanation
CRUD Admin Entity - Use Case - Users - Create, Read, Update and Delete

Listing 5: Alloy Code

```

//=========================================
//==== CRUD OPERATIONS
//==== Admin Create/Read/Update/Delete
//=========================================
pred addAdmin [a, a': Admin] {
  a' = a
}

pred showAdmin [a, a': Admin] {
  invAdmin[a]
  addAdmin[a, a']
}

pred removeAdmin[a, a': Admin] {
  a = a'
}

pred updateAdmin [a', aa': Admin] {
  a' = aa'
}

```

```
run addAdmin
run showAdmin
run removeAdmin
run updateAdmin
//========================================
```

screenshot



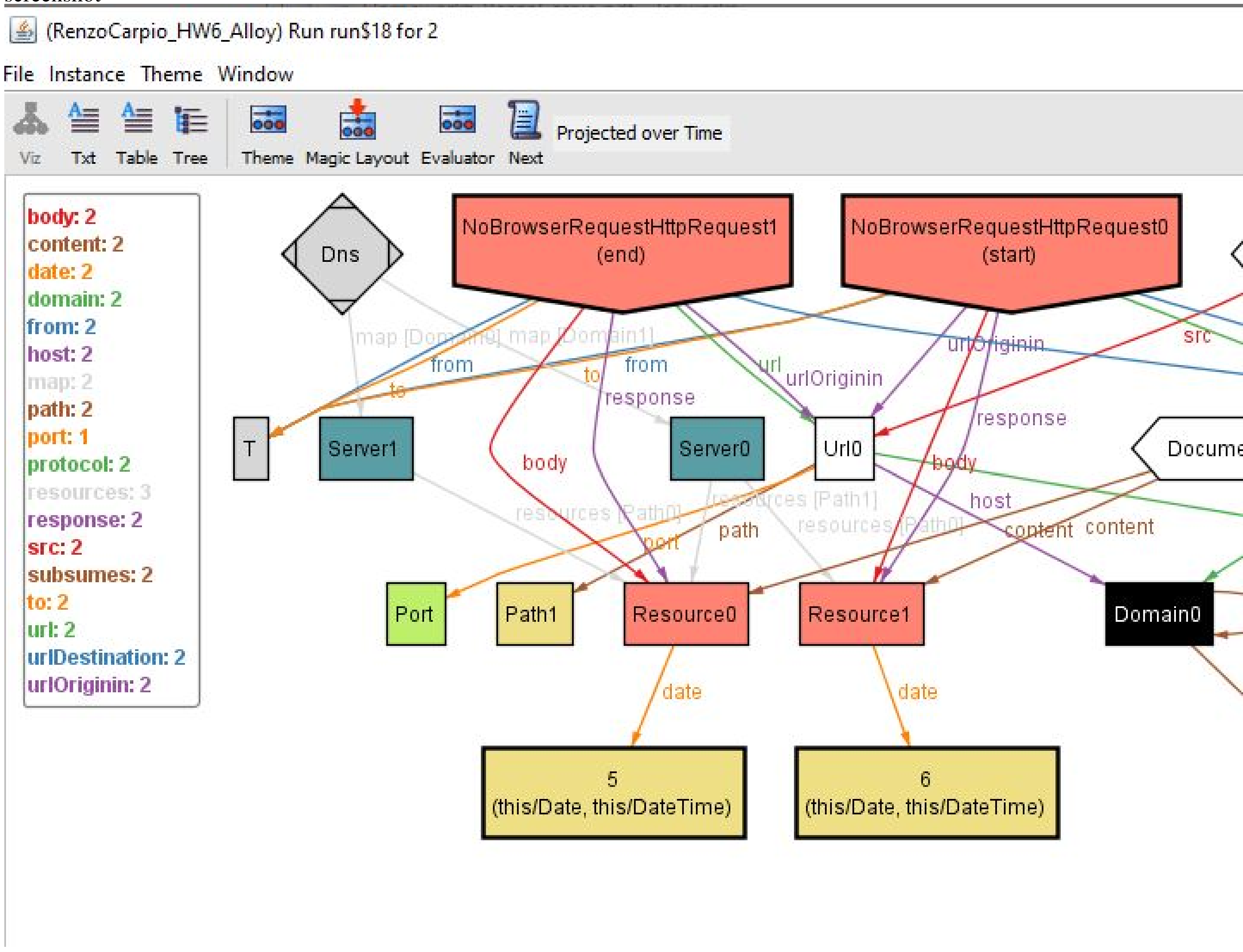
5.3 CRUD Client Request Entity

Explanation
CRUD Client Request Entity - Use Case - Users - Create, Read, Update and Delete

Listing 6: Alloy Code

```

//========================================
//==== CRUD OPERATIONS
//==== Client Request Create/Read/Update/Delete
//========================================
pred addClient [c, c': Client, bus: GenericUser ] {
  c'.browserusersession = c.browserusersession + bus
}

pred showClientCurrentSession [c, c': Client, bus: GenericUser] {
  invClientCurrentSessionn[c]
  addClient[c, c', bus]
}

pred removeEndClient[c, c': Client, bus: GenericUser ]  {
  c'.browserusersession = c.browserusersession - bus
}

pred updateClient[c', cc': Client, bus: GenericUser ]  {
    c'.browserusersession = cc'.browserusersession + bus
}

run addClient
run showClientCurrentSession
run removeEndClient
run updateClient
//======================================
```

screenshot


(RenzoCarpio_HW6_Alloy) Run run$18 for 2
File Instance Theme Window
Viz Txt Table Tree Theme Magic Layout Evaluator Next
Projected over Time
body: 2
content: 2
date: 2
domain: 2
from: 2
host: 2
map: 2
path: 2
port: 1
protocol: 2
resources: 3
response: 2
src: 2
subsumes: 2
to: 2
url: 2
urlDestination: 2
urlOriginin: 2
Dns
NoBrowserRequestHttpRequest1 (end)
NoBrowserRequestHttpRequest0 (start)
T
Server1
Server0
Url0
Port
Path1
Resource0
Resource1
Domain0
5 (this/Date, this/DateTime)
6 (this/Date, this/DateTime)

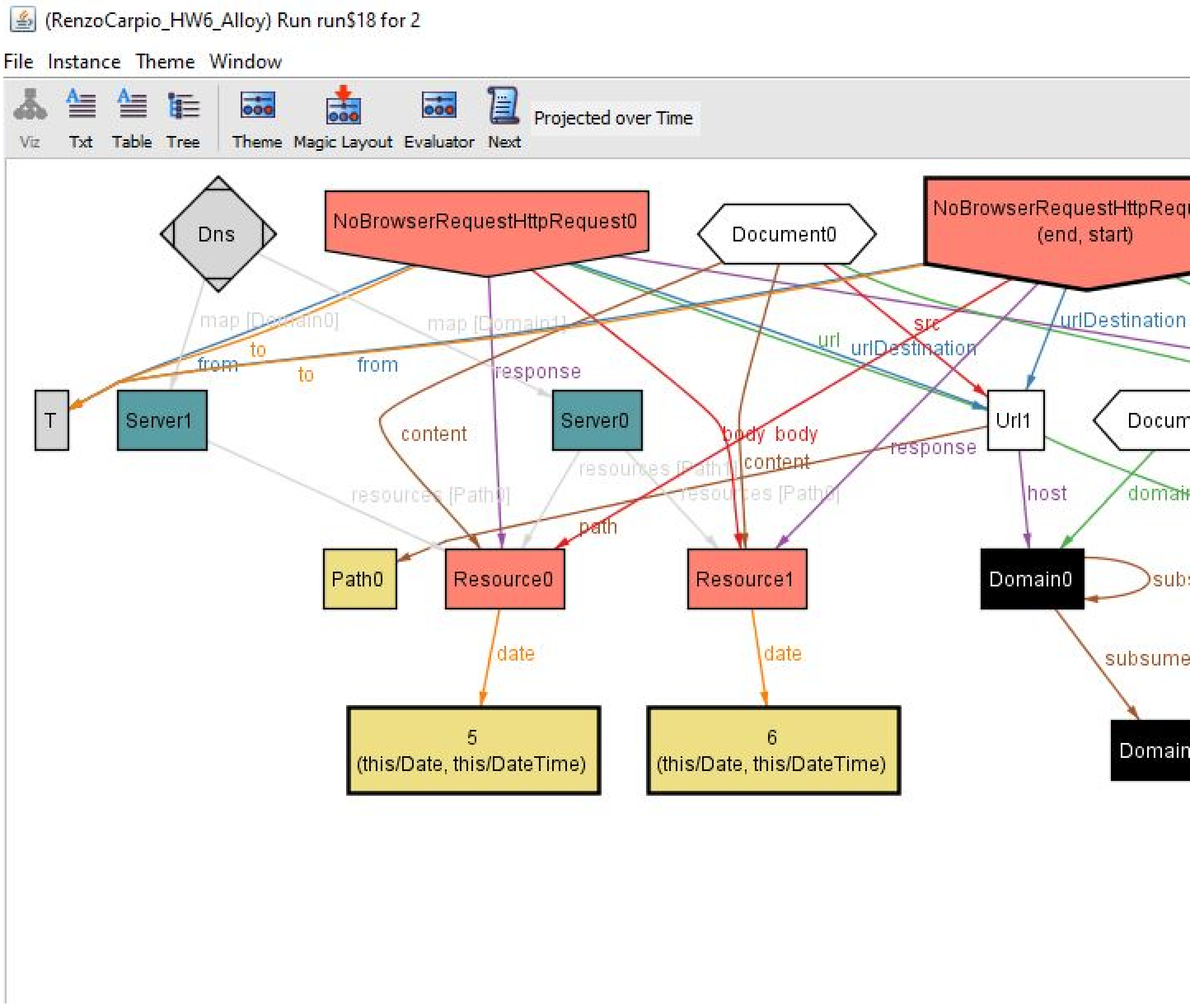
5.4 CRUD HttpRequest Request Entity

Explanation
CRUD HttpRequest Request Entity - Use Case - Users - Create, Read, Update and Delete

Listing 7: Alloy Code

```

//========================================
//==== CRUD OPERATIONS
//==== HttpRequest Request Create/Read/Update/Delete
//========================================
pred addHttpRequest [s, s': HttpRequest, res: Url] {
  s'.url = s.url + res
}

pred showHttpRequestCurrentSession [s, s': HttpRequest, res: Url] {
  invServerCurrentSessionn[s]
  addHttpRequest[s, s', res]
}

pred removeHttpRequestr[s, s': HttpRequest, res: Url ]  {
  s'.url = s.url - res
}

pred updateHttpRequest[s', ss': HttpRequest, res: Url ]  {
  s'.url = ss'.url + res
}

run addHttpRequest
run showHttpRequestCurrentSession
run removeHttpRequestr
run updateHttpRequest

//========================================
```

screenshot



6 Website Security - Transitions and Time Components

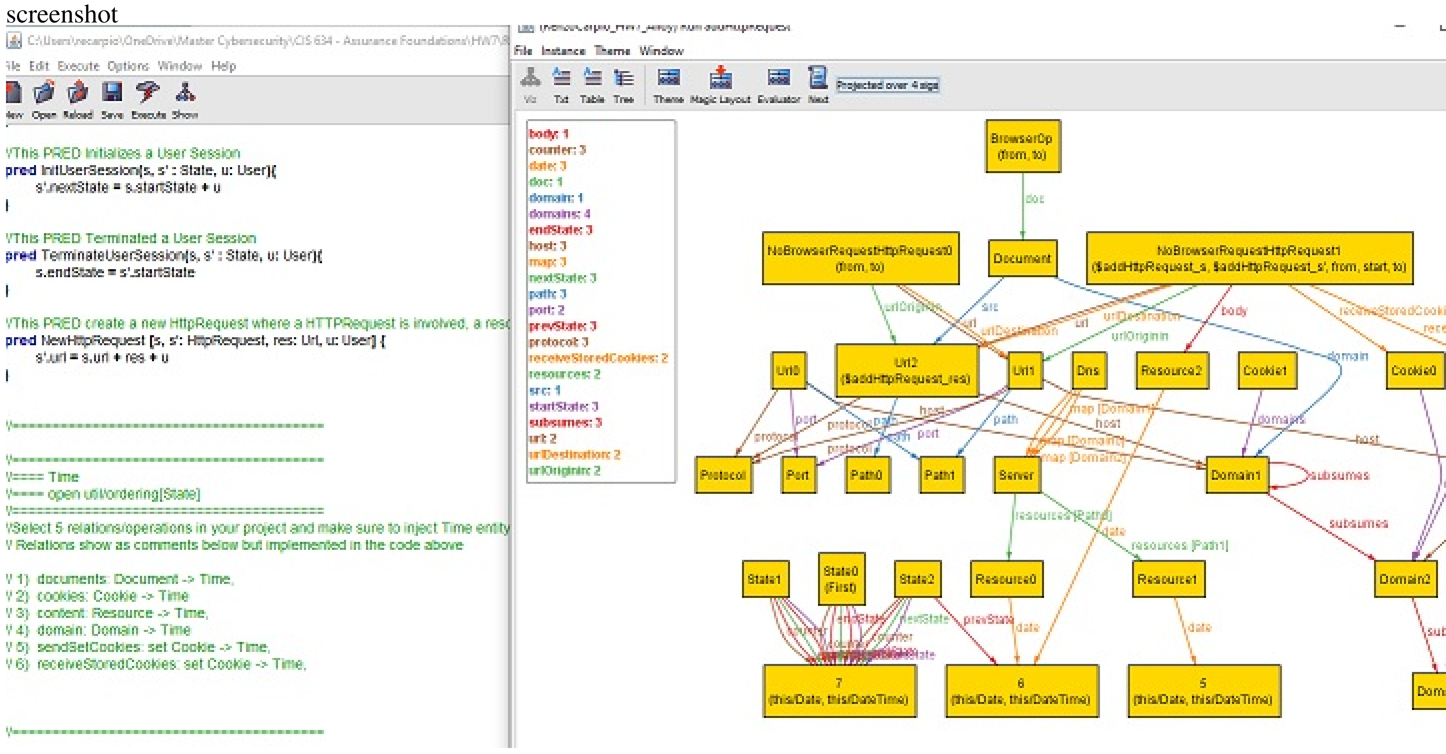
6.1 Parametric Module

Explanation
For this section, it can be observed below the usage of the Parametric module for my model
There is also a predicate that is used for setting and retrieving a cookie an acyclic operation

Listing 8: Alloy Code

```

//=========================================
//==== PARAMETRIC MODULE
//==== Oepn util/ordering
//=========================================
//Declared at the top of code
// module RenzoCarpioWebAlloySecurityPolicies[T]

//Utility being used for this sample predicate
//open util/relation as rel

//Cookies are consumed in 2 ways either  User Cient Browser is setting
//a cookie fior the first time or if a cookie exists then that cookie is
//analyzed and updated accordingly this is an  acyclic operation

pred setUpRetrievecookie[c: Cookie -> Cookie]{
  rel/acyclic [c, Cookie]
}
```

screenshot


C:\Users\recarpio\OneDrive\Master Cybersecurity\CIS 634 - Assurance Foundations\HW7\
File Edit Execute Options Window Help
New Open Reload Save Execute Show
File Instance Theme Window
Viz Txt Table Tree Theme Magic Layout Evaluator Next
Projected over 4 sigs

```
//This PRED Initializes a User Session
pred InitUserSession[s, s' : State, u: User]{
      s'.nextState = s.startState + u
}

//This PRED Terminated a User Session
pred TerminateUserSession[s, s' : State, u: User]{
      s.endState = s'.startState
}

//This PRED create a new HttpRequest where a HTTPRequest is involved, a reso
pred NewHttpRequest [s, s': HttpRequest, res: Url, u: User] {
      s'.url = s.url + res + u
}


//=========================================

//=========================================
//==== Time
//==== open util/ordering[State]
//=========================================
//Select 5 relations/operations in your project and make sure to inject Time entity
// Relations show as comments below but implemented in the code above

// 1)  documents: Document -> Time,
// 2)  cookies: Cookie -> Time
// 3)  content: Resource -> Time,
// 4)  domain: Domain -> Time
// 5)  sendSetCookies: set Cookie -> Time,
// 6)  receiveStoredCookies: set Cookie -> Time,



//=========================================
```

```
body: 1
counter: 3
date: 3
doc: 1
domain: 1
domains: 4
endState: 3
host: 3
map: 3
nextState: 3
path: 3
port: 2
prevState: 3
protocol: 3
receiveStoredCookies: 2
resources: 2
src: 1
startState: 3
subsumes: 3
url: 2
urlDestination: 2
urlOrigin: 2
```

6.2 State Modules

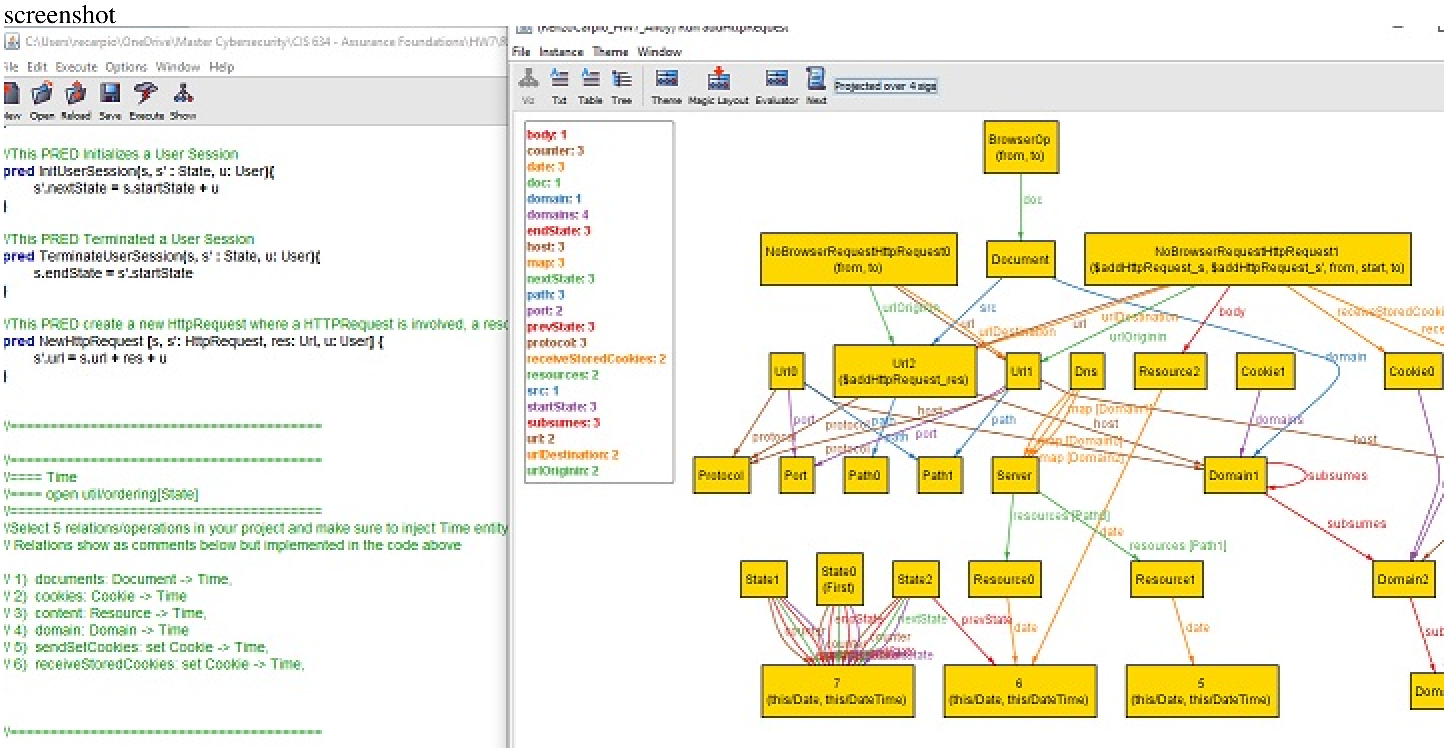
Explanation
For this section, it can be observed below the usage of the state modules for my model
There is an initial state that is defined and then 2 states are created one for initializing a UserSession
Another predicate is created to terminate a user session, changing a state from current session to next state terminated
Then a predicate is created to create a New HttpRequest, changing a request state, involving HttpRequest, Url and User

Listing 9: Alloy Code

```



//========================================
//==== OPERATIONS State
//==== open util/ordering[State]
//========================================

//This PRED Initializes a State
```

```
pred InitialState [s:State]
{
  one s.startState
  no s.nextState
}

//This PRED Initializes a User Session
pred InitUserSession(s, s' : State, u: User){
  s'.nextState = s.startState + u
}

//This PRED Terminated a User Session
pred TerminateUserSession(s, s' : State, u: User){
  s.endState = s'.startState
}

//This PRED create a new HttpRequest where a HTTPRequest is involved, a
    resource and  User
pred NewHttpRequest [s, s': HttpRequest, res: Url, u: User] {
  s'.url = s.url + res + u
}


//=========================================
```

screenshot



```
//This PRED Initializes a User Session
pred InitUserSession(s, s' : State, u: User){
      s'.nextState = s.startState + u
}

//This PRED Terminated a User Session
pred TerminateUserSession(s, s' : State, u: User){
      s.endState = s'.startState
}

//This PRED create a new HttpRequest where a HTTPRequest is involved, a res
pred NewHttpRequest [s, s': HttpRequest, res: Url, u: User] {
      s'.url = s.url + res + u
}


//=========================================

//=========================================
//==== Time
//==== open util/ordering[State]
//=========================================
//Select 5 relations/operations in your project and make sure to inject Time entity
// Relations show as comments below but implemented in the code above

// 1)  documents: Document -> Time,
// 2)  cookies: Cookie -> Time
// 3)  content: Resource -> Time,
// 4)  domain: Domain -> Time
// 5)  sendSetCookies: set Cookie -> Time,
// 6)  receiveStoredCookies: set Cookie -> Time,


//=========================================
```

```
body: 1
counter: 3
date: 3
doc: 1
domain: 1
domains: 4
endState: 3
host: 3
map: 3
nextState: 3
path: 3
port: 2
prevState: 3
protocol: 3
receiveStoredCookies: 2
resources: 2
src: 1
startState: 3
subsumes: 3
url: 2
urlDestination: 2
urlOrigin: 2
```

File Edit Execute Options Window Help
New Open Reload Save Execute Show
File Instance Theme Window
Viz Txt Table Tree Theme Magic Layout Evaluator Next
Projected over 4 sigs
BrowserOp (from, to)
NoBrowserRequestHttpRequest0 (from, to)
Document
NoBrowserRequestHttpRequest1 ($addHttpRequest_s, $addHttpRequest_s', from, start, to)
Url0
Url2 ($addHttpRequest_res)
Url1
Dns
Resource2
Cookie1
Cookie0
Protocol
Port
Path0
Path1
Server
Domain1
State1
State0 (First)
State2
Resource0
Resource1
Domain2
7 (this/Date, this/DateTime)
6 (this/Date, this/DateTime)
5 (this/Date, this/DateTime)

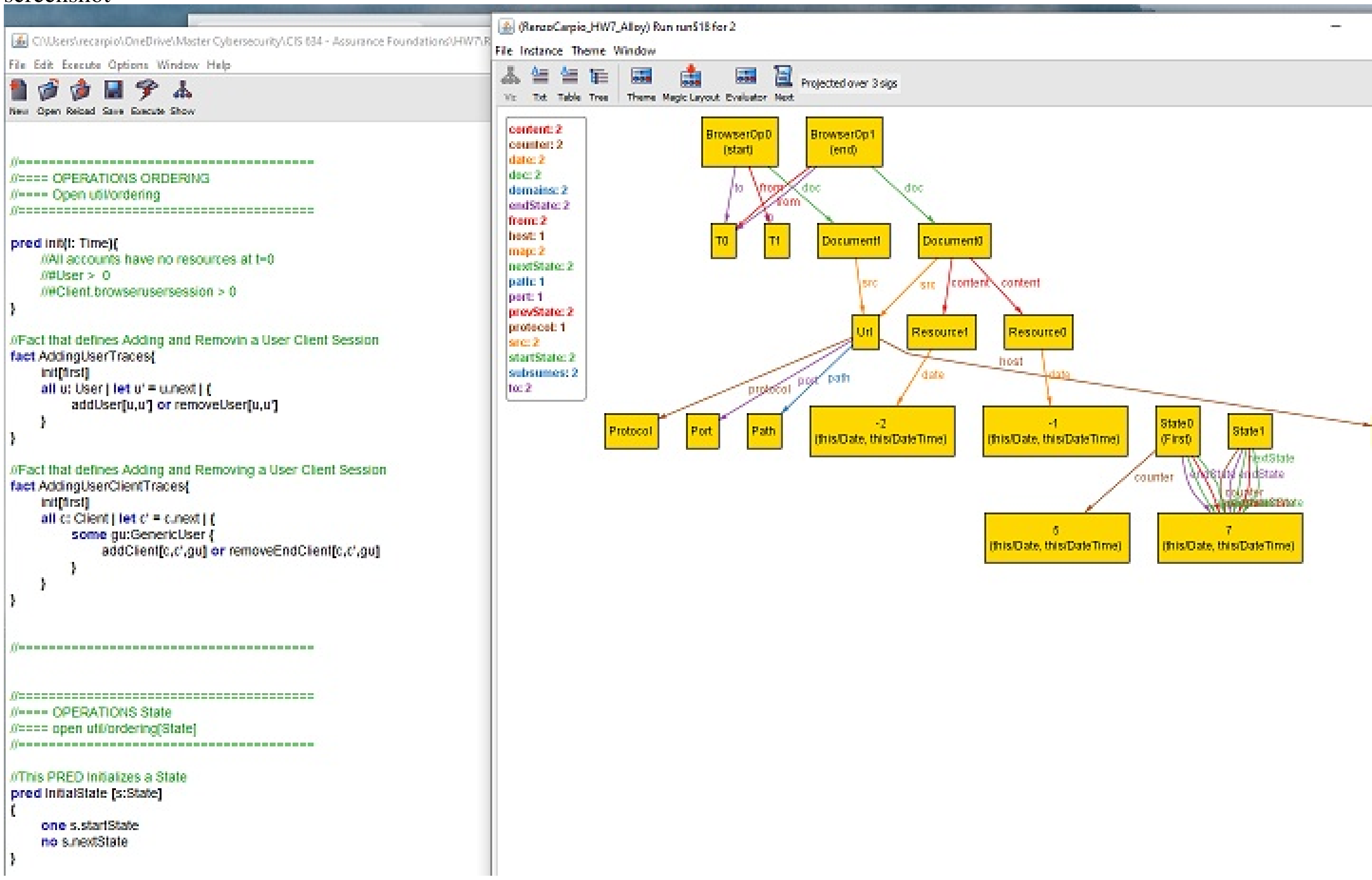
6.3 Ordering Module

Explanation
For this section, it can be observed below the usage of the ordering module for my model
Traces were used to use CRUD operations for example to add or remove a user
a trace was also used to add or remove a user from a client session
As it can be seen, the required items for this assignment were also used in the modules below pred init, fact trace, pred trans.

Listing 10: Alloy Code

```



//========================================
//==== OPERATIONS ORDERING
//==== Open util/ordering
//========================================

pred init(t: Time){
  //All accounts have no resources at t=0
  //#User >  0
  //#Client.browserusersession > 0
}

//Fact that defines Adding and Removin a User Client Session
fact AddingUserTraces{
  init[first]
  all u: User | let u' = u.next | {
    addUser[u,u'] or removeUser[u,u']
  }
}

//Fact that defines Adding and Removing a User Client Session
fact AddingUserClientTraces{
  init[first]
  all c: Client | let c' = c.next | {
    some gu:GenericUser {
      addClient[c,c',gu] or removeEndClient[c,c',gu]
    }
  }
}


//========================================
```

screenshot


C:\Users\recarpio\OneDrive\Master Cybersecurity\CIS 634 - Assurance Foundations\HW7\R
File Edit Execute Options Window Help
New Open Reload Save Execute Show
(RenzoCarpio_HW7_Alloy) Run run$18 for 2
File Instance Theme Window
Viz Txt Table Tree Theme Magic Layout Evaluator Next
Projected over 3 sigs
content: 2
counter: 2
date: 2
doc: 2
domains: 2
endState: 2
from: 2
host: 1
map: 2
nextState: 2
path: 1
port: 1
prevState: 2
protocol: 1
src: 2
startState: 2
subsumes: 2
to: 2
BrowserOp0 (start)
BrowserOp1 (end)
T0
T1
Document1
Document0
Url
Resource1
Resource0
Protocol
Port
Path
-2 (this/Date, this/DateTime)
-1 (this/Date, this/DateTime)
State0 (First)
State1
6 (this/Date, this/DateTime)
7 (this/Date, this/DateTime)
to
from
doc
src
content
date
host
protocol
port
path
counter
nextState
endState

```
//========================================
//==== OPERATIONS ORDERING
//==== Open util/ordering
//========================================

pred init[t: Time]{
	//All accounts have no resources at t=0
	//#User >  0
	//#Client.browserusersession > 0
}

//Fact that defines Adding and Removin a User Client Session
fact AddingUserTraces{
	init[first]
	all u: User | let u' = u.next | {
		addUser[u,u'] or removeUser[u,u']
	}
}

//Fact that defines Adding and Removing a User Client Session
fact AddingUserClientTraces{
	init[first]
	all c: Client | let c' = c.next | {
		some gu:GenericUser {
			addClient[c,c',gu] or removeEndClient[c,c',gu]
		}
	}
}


//========================================


//========================================
//==== OPERATIONS State
//==== open util/ordering[State]
//========================================

//This PRED initializes a State
pred InitialState [s:State]
{
	one s.startState
	no s.nextState
}
```

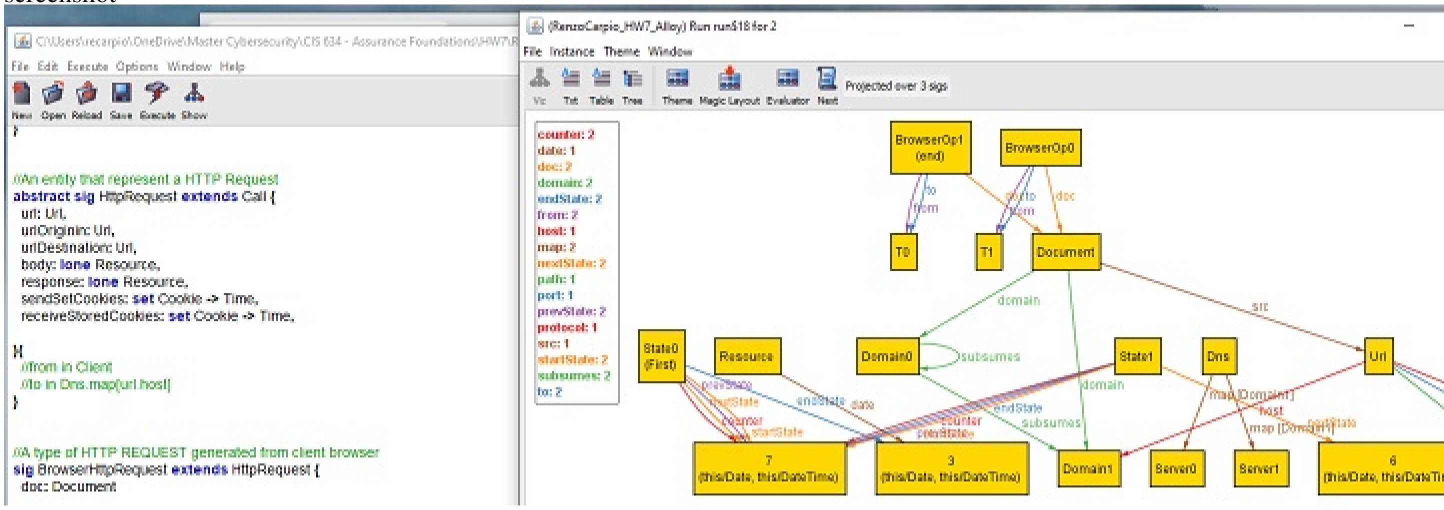
6.4 Time Entity Relationships

Explanation
The following time entities relationships are present for my model in Alloy Analyzer
Relationships below are present in the code - module //

Listing 11: Alloy Code

```

//========================================
//==== Time
//==== open util/ordering[State]
//========================================
//Select 5 relations/operations in your project and make sure to inject Time
   entity
// Relations show as comments below but implemented in the code above

// 1)  documents: Document -> Time,
// 2)  cookies: Cookie -> Time
// 3)  content: Resource -> Time,
// 4)  domain: Domain -> Time
// 5)  sendSetCookies: set Cookie -> Time,
// 6)  receiveStoredCookies: set Cookie -> Time,



//========================================
```

screenshot



C:\Users\recarpio\OneDrive\Master Cybersecurity\CIS 634 - Assurance Foundations\HW7\R
File Edit Execute Options Window Help
New Open Reload Save Execute Show
(RenzoCarpio_HW7_Alloy) Run run$18 for 2
File Instance Theme Window
Viz Txt Table Tree Theme Magic Layout Evaluator Next
Projected over 3 sigs
BrowserOp1 (end)
BrowserOp0
T0
T1
Document
Domain0
State0 (First)
Resource
State1
Dns
Url
7 (this/Date, this/DateTime)
3 (this/Date, this/DateTime)
Domain1
Server0
Server1
6 (this/Date, this/DateTi

```
}


//An entity that represent a HTTP Request
abstract sig HttpRequest extends Call {
  url: Url,
  urlOriginin: Url,
  urlDestination: Url,
  body: lone Resource,
  response: lone Resource,
  sendSetCookies: set Cookie -> Time,
  receiveStoredCookies: set Cookie -> Time,

}{
  //from in Client
  //to in Dns.map[url.host]
}


//A type of HTTP REQUEST generated from client browser
sig BrowserHttpRequest extends HttpRequest {
  doc: Document
```

```
counter: 2
date: 1
doc: 2
domain: 2
endState: 2
from: 2
host: 1
map: 2
nextState: 2
path: 1
port: 1
prevState: 2
protocol: 1
src: 1
startState: 2
subsumes: 2
to: 2
```

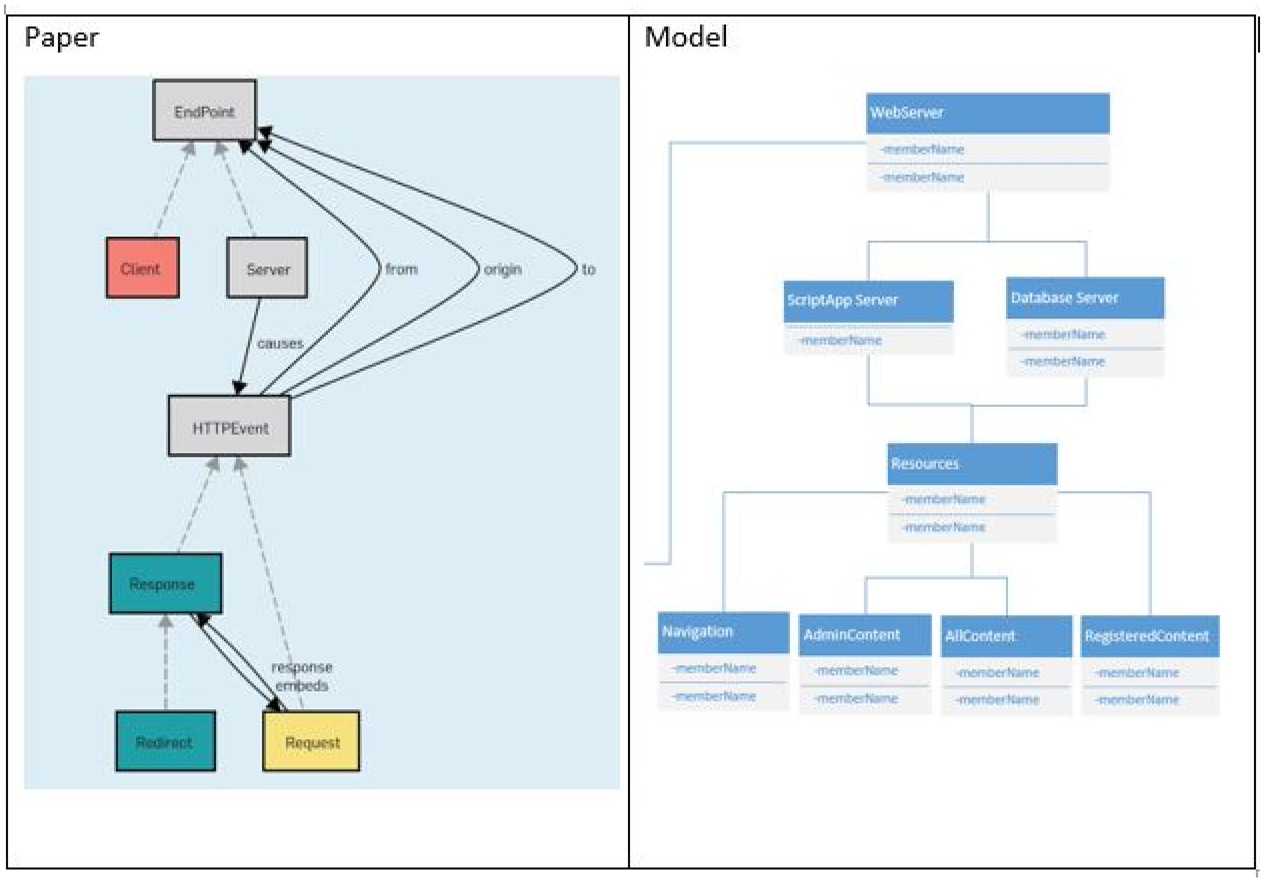
7 Research Papers Related Work

7.1 Paper 1 - Alloy: a language and tool for exploring software designs.

`https://dl.acm.org/doi/abs/10.1145/3338843`

T his paper starts analyzing and describing the purpose of Alloy, in this specific paper it is mentioned that Alloy is a language and a set of tools that allows the exploration of many software designs and inside of these designs the many structures that are part of the model either seen as a static model or as a dynamic model given the rules to which the model needs to comply. This paper focus in the advantages of modeling and designing software in an agile way. In each iteration of the software design, it is very helpful to have a representation and a way to analyze the different constrains and rules that need to be enforced and that only pertain to that software. Also, this article describes the tools that were used before Alloy when trying to analyze and model a software design problem, the two tools that this article describe are the Theorem Provers and Model Checkers. Based on the limitation of these previous tools for modeling design, Alloy was born and the core advantages that Alloy had compared with the other tools was that Alloy approached the problem using: Relational logic, models have entities, and these entities have relationships among them. Small scope analysis, where analysis perform in software is no longer a competitive advantage in the long term but a major investment beyond modeling which is crucial nowadays when creating software. Translation to SAT, Alloy uses tactics used by SAT to reduce problem solving by adding constrains in a model and by trying to break the symmetry in some models which could lead to foresee problems in the design of software development.

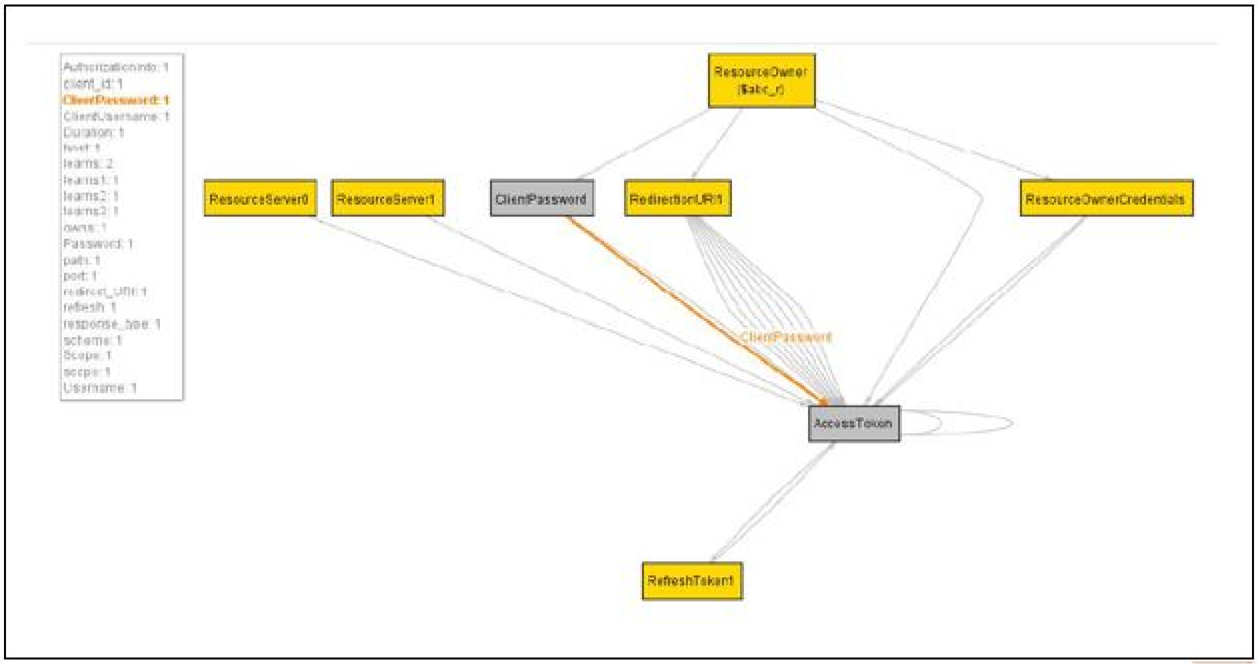
Comparison to my model "Alloy – Website Security Policies" This paper describes an example of a known attack which is called cross site forgery, in this type of attacks a end user is forced or unseemly unaware to execute malicious code in the background, the user could be either authenticated or not.
As it can be seen in the graphic above, the left side of the table contains a scenario showing the interaction of a web request, in the paper it highlights what entities are present in the model and it also shows the interaction between these entities. A web request could possess a flaw and be exposed to a CSRF exploitation. In this case the paper showcases different checks using Allow to try to prevent this scenario. In my case and given my project, "Website Security Policies" some of these examples can be applied to my project, in reality the whole web request from the endpoint, client browser in my case, all the way to the response, in my case access to resources. Different constraints are described in the paper that could be applied to my project: enforce an origin, require a response, request direction, etc.



7.2 Paper 2 - Formal Verification of OAuth 2.0 using Alloy Framework.

`https://ieeexplore.ieee.org/abstract/document/5966531`

The purpose of this paper is to explain the use of OAuth 2.0 protocol and how to successfully discover any vulnerabilities that could be hidden using Alloy Analyzer. As an introduction this paper explains in general terms the term OAuth, this protocol can be defined as the de facto protocol that aims to create a secure method without using a user/service account password. Before explaining how Alloy is used in this project the paper tries to explain the functionality and flow of OAuth protocol. There are 4 parts involved: Resource Owner: which is the entity that has the right to provide permissions to certain resources for which it has access. Resource Server: which is the unit in charge to protect resources. It also controls the access to these resources using tokens. Client: which is the entity to makes the request to the resources. Authorization Server: which is the entity in charge of granting tokens to the clients after a successfully authentication. In this paper, Alloy Analyzer is used to show the security vulnerability that OAuth could have when the client password lives in client desktop app which can be vulnerable to attacks by using reverse engineering and later using the compromised secret client credential to create a malicious application that uses this hacked credential.



Comparison to my model "Alloy – Website Security Policies" Similar to the first paper that was summarized in the previous section, this paper is very useful in my current project. Most websites have authentication modules for users. We have 3 entities: regular users, registered/authenticated users and administrators. While modeling and creating policies for websites it is especially important to pay attention in the authentication module which is crucial to keep resources safe, and only to provide right access to authenticated users. In this paper a possible flow was found in the process for OAuth authentication protocol which can be observed in the previous graphic. A counter example, in the Alloy Analyzer mode, was found using the entities explained previously. In which a resource can be accessed using a clients secret key but the protocol tries to prevent the access to these resources.

7.3 Paper 3 - Design and validation of a general security model with the alloy analyzer.

`https://www.researchgate.net/publication/228577321_Design_and_validation_of_a_general_security_model_with_the_alloy_analyzer`

T his paper touches a broader topic in regards security of an application in general, Alloy model in this paper tries to analyze the security of various network protocols, but it can be easily applied to a web request and the different requirements that are needed to provide security to a web application. This paper explains different topics regarding security and how those topics can be modelled using Alloy Analyzer. The Alloy models created in this paper are quite extensive and show different constrains, pre-conditions, post-conditions, etc. The examples provided in this paper are quite advanced in terms of Alloy modeling some models in the paper even include the use of Time entity to present security vulnerabilities in certain "projections" in time of the model. In this paper different logical systems of security were used to define the security model: BAN Logic, Knowledge Logic S5, Communication Models. And Alloy was used to validate all the definition of security and the computability among them. The security model presented in this paper has 3 main entities: Processes- which are the agents in the model. Formula- which is the data. Policy- which defines if the process entity has access to the formula entity.

Some of the security definitions tested with the Alloy model were: Confidentiality only people with the proper access will have access to the data. Integrity check that the request to access the data is valid. Authenticity if there is a valid request check level of access to the protected resource. The model created in Alloy and which is explained in this paper tries to prevent some of the obstacles that are usually exposed for the entities explained before. Eavesdropping a process entity is listening to the contents of a protected message even though this entity did not have access to that specific resource content. Corruption some process which does not have write access to an assigned resource. Spoofing a particular process is not the real source of origin of a protected message. In the model that was created in Alloy, the paper presented different instances to validate the obstacles of security mentioned before. The model was able to also generate some instances where some violations were generated for certain security definitions.

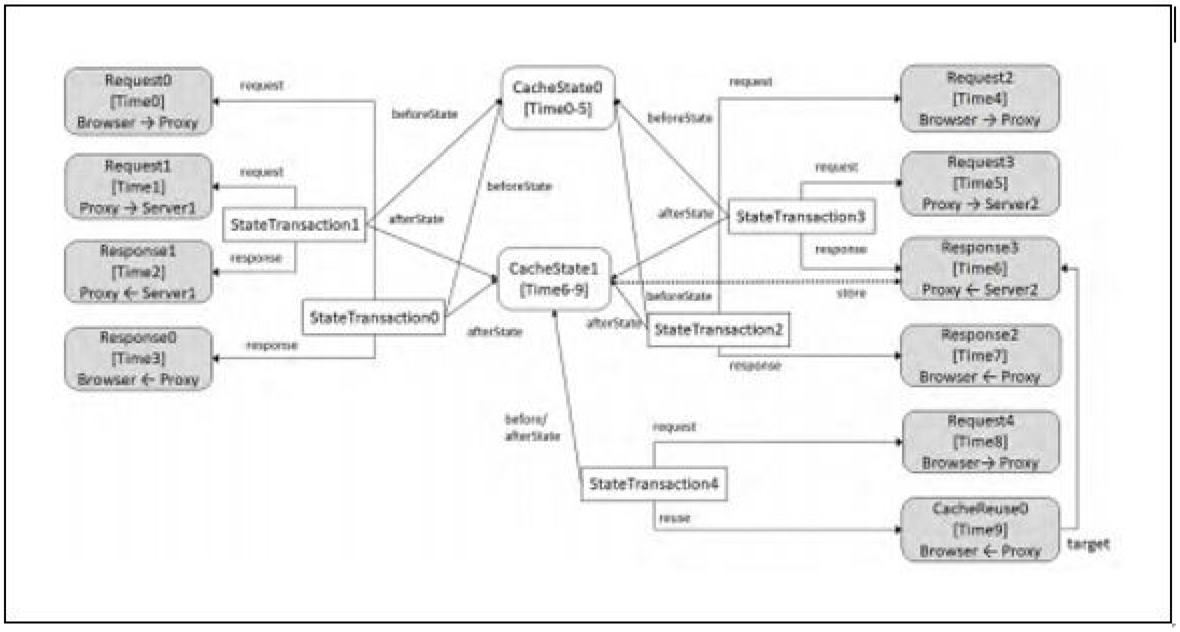
Comparison to my model "Alloy – Website Security Policies" The main contribution of this paper in general for any project that we are discussing this term in our class are the different security definitions that are included and are tackle inside Alloy Analyzer model, that is presented in this paper. In my particular scenario for "Website Security Policies" there are many constrains, pre-conditions, post-conditions, etc.; that can be used to create a model which will prevent many security flaws in a website. For example, the check for eavesdropping is crucial when talking about security policies. This instance should be covered when trying to model security policies in a website. In my model there can be check of Confidentiality in a web request, which is been shown in this paper. When a user starts a request to access a resource in the website that user request has to be validated in its integrity to avoid some of the security flaws explained before. In this case this model uses 3 main entities as explained before. Processes, Formula and Policy. In my case it can be the equivalent of ClientBrowser, WebServer and Resource which are the entities in Alloy model.

7.4 Paper 4 - Towards Further Formal Foundation of Web Security: Expression of Temporal Logic in Alloy and Its Application to a Security Model With Cacher.

`https://www.researchgate.net/publication/228577321_Design_and_validation_of_a_general_security_model_with_the_alloy_analyzer`

T his paper explains some formal methods to conduct security analysis on websites. What is particularly interesting in this paper is that it presents a new web security model. Current models have some limitations while trying to express parallel communication and storage needs such as caches of the sites. Currently Alloy Models do not have a way to present or create a model which has a temporal logic". This paper tries to overcome this problem with a proposition where time and parallel access are taken in consideration to be able to be applied to a security model that allows the modeling of different security scenarios that cannot be contemplated using other more complicated techniques. This prosed new technique/model that can be used on Alloy takes in consideration not only time but also multiple access to the same resource at the same time. And in the paper the author coined the terms Temporal Logicto analyze security scenarios where a model needs to test the access of multiple sessions in parallel to access the same or different services. Temporal logic is a limitation of Alloy Analyzer, but this paper prosed an addition to Alloy analyzer for these situations. This new syntaxis allows testing different security models that uses cache, time, and parallel access. In this case the paper tries to explain some behaviors of the cache and how the cache can be misused for certain attacks. The Alloy Analyzer model extension as explained previously allows to take in consideration different factors that traditional models in Alloy do not contemplate. Some of the case of studies that are approached in the paper to model cache security and events to prevent flaws and attacks that uses cache and cookies are the following: Reuse of stored responses- how cache and cookies are present in a web request and the different states that each goes through during the different times in the request, Alloy model tries to illustrate and prevent this types of attacks. Verification Stored Responses- This is the response from a server, a server provides cache result or a new result, this response could be subject to an attacker in a given window of time, paper tries to model how to prevent this attack. Same origin attack which are most know as cache poisoning attack. The Alloy model presented for this topic tries to prevent same-origin browser cache poisoning, a man in the middle attack, eavesdropping information or tampering information for attacking purposes. The model shows the different transitions of a requests and also the states of the request. Other attacks that are approached in this paper are: forgery attacks, cross site bad requests , cross origin browser cache and phising attacks

Comparison to my model "Alloy – Website Security Policies" As it was explained this extension of Alloy, this new syntax, allows the creation of a model to perform a security investigation of the web given different scenarios where time, state, parallelism are involved. In the case of my model this is an advanced topic but to extend the case study of my model this will be a great starting point because based in the previous description it will allow the creation of a stronger model which has in consideration multiple scenarios with the characteristics mentioned before.
My model could easily use this new syntaxis while creating a security policy to check the security for a possible "cross-origin browser cache poisoning attack".


Request0 [Time0] Browser → Proxy
Request1 [Time1] Proxy → Server1
Response1 [Time2] Proxy ← Server1
Response0 [Time3] Browser ← Proxy
StateTransaction1
StateTransaction0
CacheState0 [Time0-5]
CacheState1 [Time6-9]
StateTransaction3
StateTransaction2
StateTransaction4
Request2 [Time4] Browser → Proxy
Request3 [Time5] Proxy → Server2
Response3 [Time6] Proxy ← Server2
Response2 [Time7] Browser ← Proxy
Request4 [Time8] Browser→ Proxy
CacheReuse0 [Time9] Browser ← Proxy
request
response
beforeState
afterState
before/ afterState
store
reuse
target

7.5 Paper 5 - Role-Based Access Control Modeling and Validation.

`https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6754925`

T his paper tries to explain role based access control, some examples are used to describe different security models to check consistencies and inconsistencies that are present in the model. The core components that are present in this paper and the model that it describes are subjects (users, processes) and objects (data, programs) and how is the interaction between subject and objects while taking in consideration a defined set of rules which defines the model security policies. The security model that is present in this paper tries to create a model that will allow the creation of consistent system when role base access control is needed. To explain Role-Based access control this paper uses 3 rules: Role assignment, Role Authorization and Transaction Authorization. This paper present two tables the first table has the different roles of the system and the second table contains the different transactions that can be performed by each role. The model in this paper has 4 main sections. The first section defines all the roles and the transactions. The second section define table number two which contains all the transactions available. And the section number three of the model specifies all the tasks that can be performed by a specific role. Section 4 creates different rules in the model based in details of section 3. Once that all these sections have been implemented in Alloy, the paper shows a final model which helps with checking the system validity taking in consideration different roles and the actions that can or cannot perform based in the policies defined by the model.

Comparison to my model "Alloy – Website Security Policies" Some predicates can be used from this paper in my model to check access for certain users to some restricted contents based on their role. For example, my model contains an Admin entity which is the only entity that has access to Admin content. This role-based access control paper can help create some constrains based in the role of the users.

8 References

- Alloy: a language and tool for exploring software designs.
  `https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6754925`
- Formal Verification of OAuth 2.0 using Alloy Framework.
  `https://ieeexplore.ieee.org/abstract/document/5966531`
- Design and validation of a general security model with the alloy analyzer.
  `https://www.researchgate.net/publication/228577321_Design_and_validation_of_a_general_security_model_with_the_alloy_analyzer`
- Towards Further Formal Foundation of Web Security: Expression of Temporal Logic in Alloy and Its Application to a Security Model With Cache
  `https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8730354`
- Role-Based Access Control Modeling and Validation
  `https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6754925`