



Need More Sleep? REST Could Help

Drew Branch, Security Analyst I
dbranch@securityevaluators.com

Introduction

Drew Branch, CEH

Associate Security Analyst @ ISE

B.S. Electrical/Computer Engineering

M.S. Cybersecurity

Hobbies: Learning about new tech and
playing sports

Introduction

ISE

- Based in Baltimore
- High end custom security assessments
- Assess new web technologies

Overview

- REST Background
- REST vs SOAP
- REST Concepts
- Common Security Mistakes
- Hot to Fix

Background

- REpresentational State Transfer (REST)
- Defined by Roy Fielding in 2000
 - Also one of the main contributors of the HTTP specification

REST vs SOAP

REST

- No specification guidelines
- Easier to create documentation
- Scalable
- Makes use of HTTP Methods

SOAP

- Uses SOAP protocol to exchange data
- Bound by SOAP specification
 - Break one requirement, API is not SOAP
- Uses HTTP POST method

REST vs SOAP

Sample Requests

REST Request

```
POST /auth HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Content-Length: 49
Cache-Control: no-cache
Origin:
chrome-extension:
//fhhbjgbiflinjbdggehcdcbncdddodomop
User-Agent: Mozilla/5.0
(Macintosh; Intel Mac OS X 10_11_3)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/53.0.2785.116 Safari/537.36
Content-Type: application/json
Accept: */*
DNT: 1
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8

{
  "username": "admin",
  "password": "admin"
}
```

SOAP Request

```
POST /auth HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Content-Length: nnn
Cache-Control: no-cache

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap=
"http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle=
"http://www.w3.org/2003/05/soap-encoding">
<soap:Body xmlns:m=
"http://www.example.org/stock">
  <m:GetStockPrice>
    <m:StockName>IBM</m:StockName>
  </m:GetStockPrice>
</soap:Body>
</soap:Envelope>
```

HTTP Methods

- GET – Retrieve a resource
- POST – Create a new resource
- DELETE – Delete a resource at specified URI
- PATCH – Modify the resource; not replace
- PUT – Replace resource with a newly-updated representation
 - Can also create resources
- Other methods include: OPTIONS, HEAD, and TRACE

Status Codes

Code	Description	Code	Description
200	OK	401	Unauthorized
201	Created	403	Forbidden
202	Accepted	404	Not Found
400	Bad Request	500	Internal Server Error

Data Format

No rules on the format of data

Q. How does client/server know what format the data is in?

A. Header Content-Type

- text/xml
- application/json

Data Format

Content Negotiation

Accept Header

- Client sends Server header of preferred data type

Server returns data in preferred format or returns error if data type is not supported

Resource URIs

Static resource URI

- `www.example.com/blogname`
 - Each blog has a static webpage

RESTful URI

- `www.example.com/blogs/{blogId}`
 - URI stays the same if web application is changed
 - Independent of framework

Resource Relationships

Better to group resources that belong to another resource in a subfolder instead of its own folder

Makes it clear that a resource belongs to a particular resource

Resource Relationships

`www.example.com/comments/{commentId}`

- Treats both blogs and comments as separate entities
- Loses relationship between a blog and its comments

`www.example.com/blogs/{blogId}/comments/{commentId}`

- Keeps relationships

HATEOAS

Hypermedia
As
The
Engine
Of
Application
State

HATEOAS

Sample response from server containing URIs to:

- The blog's comments
- The authors profile

GET /blog/1

```
{  
  "id": "1",  
  "author": "dbr@n",  
  "date": "15June2016",  
  "commentsUri": "api/blogs/1/comments/",  
  "authorProfileUri": "api/profiles/1"  
}
```

RESTful API Classification

Is my API fully RESTful? Almost RESTful?
Or not at all

Classification is based off of the Richardson
Maturity Model

RESTful API Classification

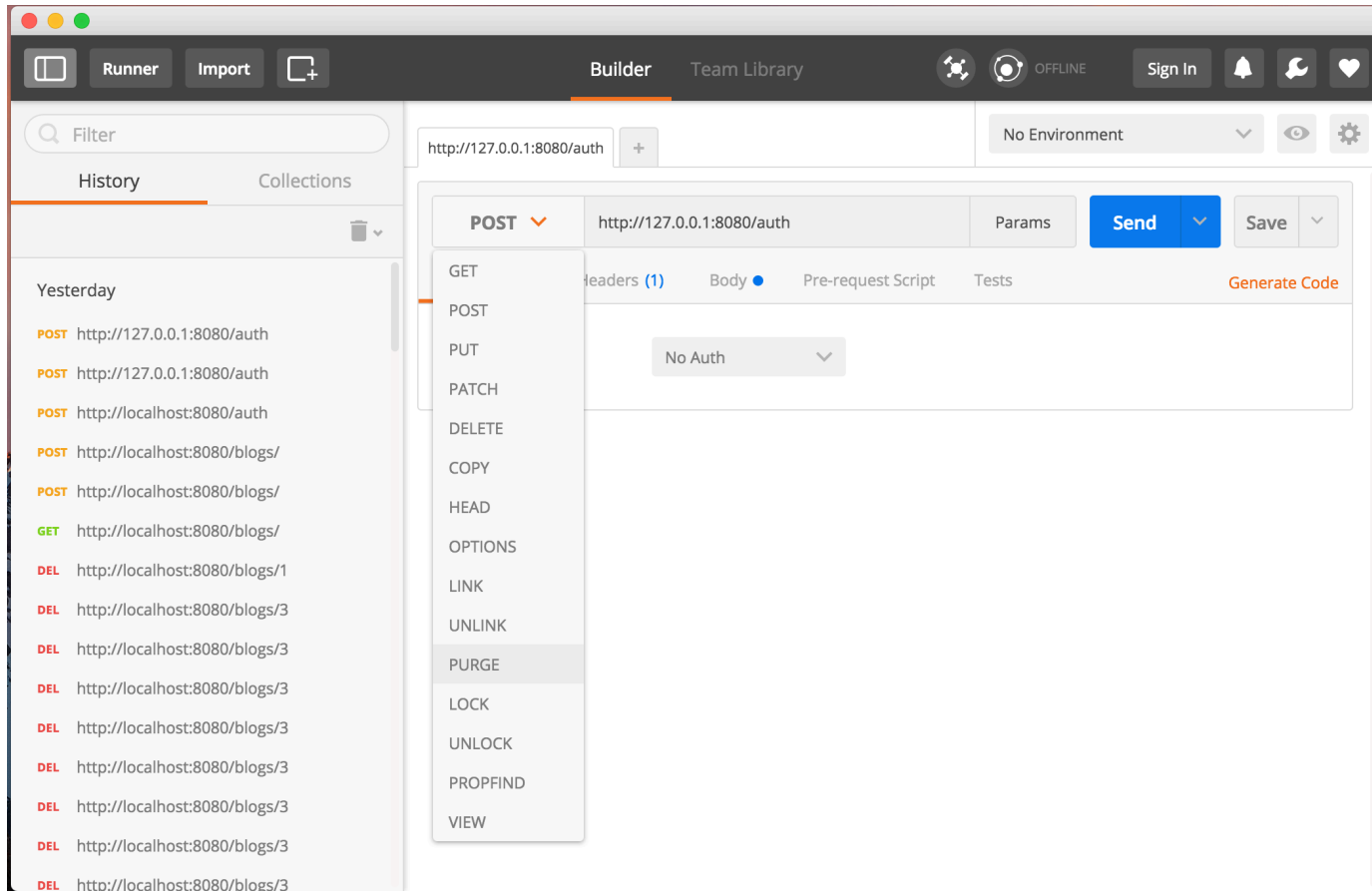
- Level 0 - Not RESTful at all
- Level 1 - Use of resources
 - Individual URI for each resource
 - /profiles/{id}
 - /blogs/{id}
 - /blogs/{id}/comments/{id}
 - Request will still contain operation
- Level 2 – Use of HTTP methods and status codes
- Level 3 - Hypermedia controls (HATEOAS)

Testing RESTful APIs

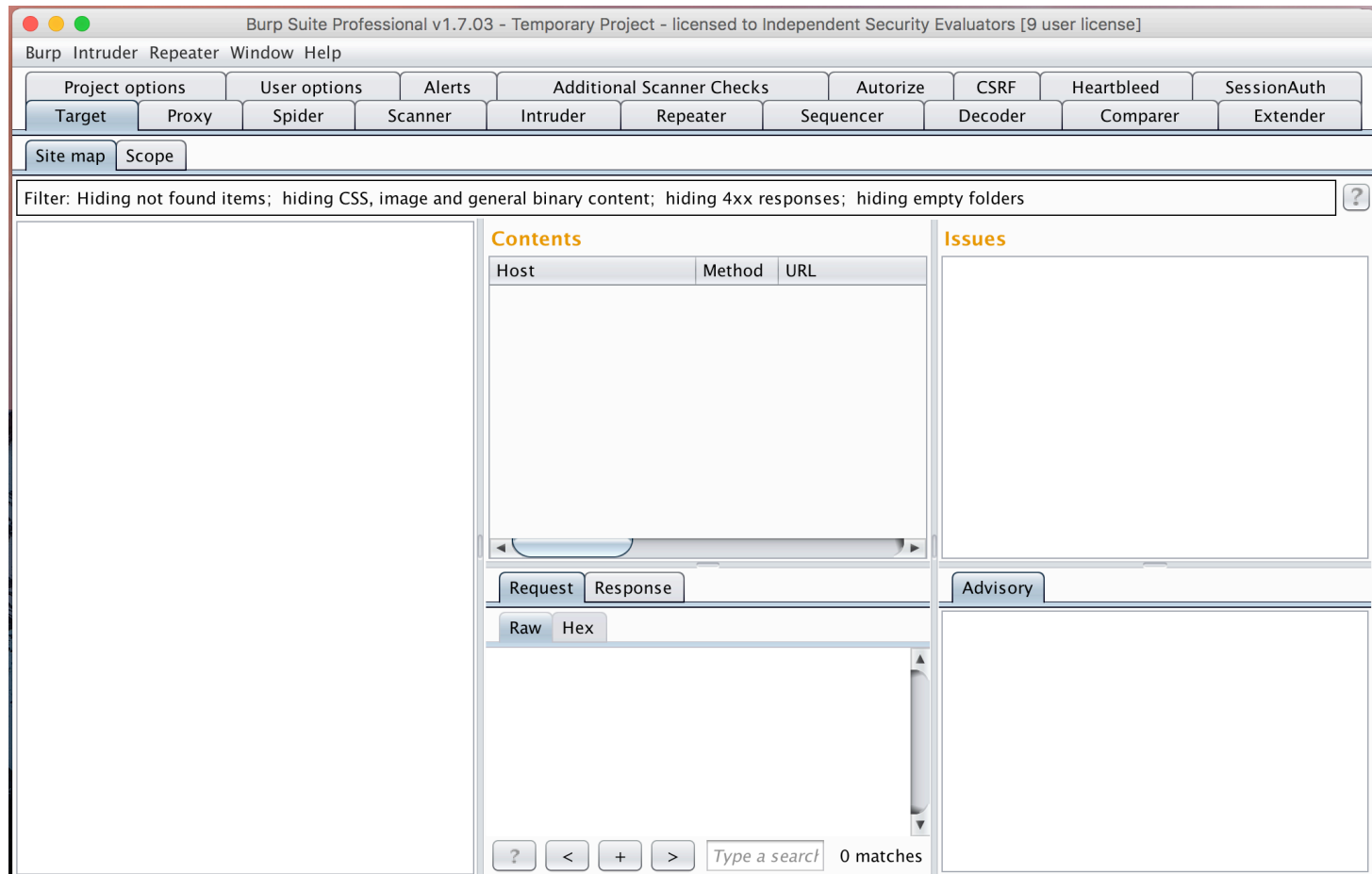
Useful Tools

- Use Postman to test during development
- Use Burp Suite to assess APIs during assessment

Postman



Burp Suite



Security Concerns

- Unauthenticated/Unauthorized modification of “protected” assets
- Unauthenticated/Unauthorized access to “protected” assets
- Replay Attacks

Sample API

Created a simple blog RESTful API

- Eclipse (MARS.1)
- Spring
- Tomcat v8

Protect HTTP Methods

- Not every method is valid for every resource
- Whitelist allowable methods
- Do not allow delete for critical files

Protect HTTP Methods

Annotate endpoints with allowable methods

```
@RequestMapping(path = "/blogs", method = RequestMethod.GET)
public List<Blog> getAllBlogs(){
    return new ArrayList<Blog>(blogs.values());
}
```

Spring

Protect HTTP Methods

Annotate endpoints with allowable methods

```
@Path("/blogs")
public class BlogResource {

    BlogService blogService = new BlogService();

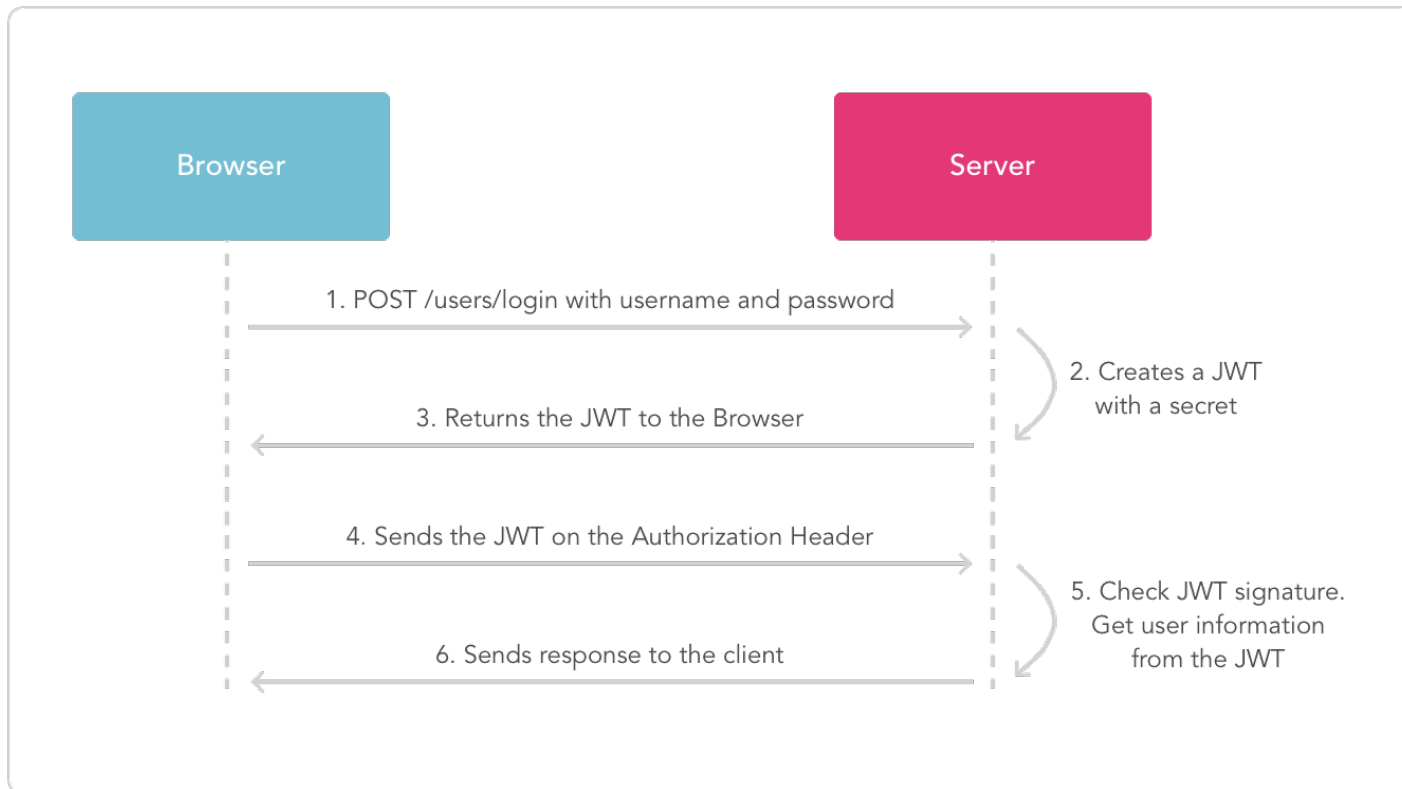
    @GET
    @Produces(MediaType.APPLICATION_JSON)
    public List<Blog> getBlogs(){
        return blogService.getAllBlogs();
    }
}
```

Jersey

JSON Web Tokens (JWT)

- Used to verify the sender
- Self contained (Stateless)
- In JSON format
- Base64 encoded
- Cannot be secured using HTTP cookie flags
- Sent within request's Authorization header
 - Primarily used for authentication but could be utilized for authorization

JSON Web Tokens (JWT)



Reference: <https://jwt.io/introduction/>

JSON Web Tokens (JWT)

Made up of three parts:

- Header
- Payload
- Signature

Payload contains claims

- Statement about entity (usually a user)
- Metadata about token

JSON Web Tokens (JWT)

Sample JWT

header

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

payload

```
{  
  "iss": "dbr@n",  
  "sub": "userName",  
  "exp": 1426420800  
}
```

signature = HS256(base64(header) + "." + base64(payload))

Format: header.payload.signature

eyJhbGcxMiJ9.eyJzdWIiOiJ1XVka9.caf1R0kof9V5b2019

JSON Web Tokens (JWT)

- Should not contain sensitive information
 - not encrypted only base64 encoded
- JWT could be encrypted using JSON Web Encryption (JWE)
- Secret used to create signature should be secured server-side

JSON Web Tokens (JWT)

Securing Endpoints with JWT

- Verify signatures
- Set short expiration dates
- Communicate over HTTPS
- Secure signing secret server side
- Do not include sensitive data in JWT
 - If so, use JWE

JSON Web Tokens (JWT)

Securing Endpoints with JWT

Target: http://localhost:8080

Request

```
GET /blogs HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Content-Length: 0
Cache-Control: no-cache
```

Response

```
HTTP/1.1 200
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Type: application/json;charset=UTF-8
Date: Tue, 20 Sep 2016 15:48:52 GMT
Content-Length: 285

[ {
  "id" : 1,
  "blog" : "Hi",
  "created" : 1474386353103,
  "author" : "dbran"
}, {
  "id" : 2,
  "blog" : "This is a dope blog",
  "created" : 1474386353103,
  "author" : "admin"
}, {
  "id" : 3,
  "blog" : "Another dope blog",
  "created" : 1474386353103,
  "author" : "user"
} ]
```

588 bytes | 11 millis

JSON Web Tokens (JWT)

Request Filter to intercept all requests and verify JWT token

```
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
throws IOException, ServletException {

    HttpServletRequest httpRequest = (HttpServletRequest) request;
    String authToken = httpRequest.getHeader(this.tokenHeader);
    String username = jwtTokenUtil.getUsernameFromToken(authToken);

    if (username != null && SecurityContextHolder.getContext().getAuthentication() == null) {
        UserDetails userDetails = this.userService.loadUserByUsername(username);
        if (jwtTokenUtil.validateToken(authToken, userDetails)) {
            UsernamePasswordAuthenticationToken authentication = new
UsernamePasswordAuthenticationToken(userDetails, null, userDetails.getAuthorities());
            authentication.setDetails(new
WebAuthenticationDetailsSource().buildDetails(httpRequest));
            SecurityContextHolder.getContext().setAuthentication(authentication);
        }
    }

    chain.doFilter(request, response);
}
```

JSON Web Tokens (JWT)

JSON Validate Method

```
public Boolean validateToken(String token, UserDetails userDetails) {  
    JwtUser user = (JwtUser) userDetails;  
    final String username = getUsernameFromToken(token);  
    final Date created = getCreatedDateFromToken(token);  
  
    return (  
        username.equals(user.getUsername())  
        && !isTokenExpired(token)  
        && !isCreatedBeforeLastPasswordReset(created,  
            user.getLastPasswordResetDate()));  
}
```

JSON Web Tokens (JWT)

Burp Suite Professional v1.7.04 - Temporary Project - licensed to Independent Security Evaluators [9 user license]

Target: http://localhost:8080

Request

```
GET /blogs HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Content-Length: 4
Cache-Control: no-cache
```

Response

```
HTTP/1.1 401
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Type: application/json;charset=UTF-8
Date: Tue, 20 Sep 2016 19:51:24 GMT
Content-Length: 130

{
  "timestamp" : 1474401084179,
  "status" : 401,
  "error" : "Unauthorized",
  "message" : "Unauthorized",
  "path" : "/blogs"
}
```

433 bytes | 5 millis

Protect Sensitive Resources

- Implement access controls
 - Role based and/or user-level controls
- Every user/tenant should not have access to every resource

Protect Sensitive Resources

Go Cancel < >

Target: <http://localhost:8080> ?

Request

Raw Params Headers Hex

```
POST /auth HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Content-Length: 51
Cache-Control: no-cache
Content-Type: application/json

{
  "username": "user",
  "password": "password"
}
```

? < + > Type a search term 0 matches

Ready

Response

Raw Headers Hex

```
HTTP/1.1 200
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0,
must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Type: application/json;charset=UTF-8
Date: Tue, 20 Sep 2016 15:46:55 GMT
Content-Length: 222

{
  "token" :
  "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0Ij0iMTY5MjUwNjYvDK9B1F9Zm7gLDMBTnayuaCJb0UK7G7sBxLm7
  NZceaGLPVWh7dAozPoZ3A"
}
```

? < + > Type a search term 0 matches

525 bytes | 457 millis

Protect Sensitive Resources

Target: http://localhost:8080

Request

Raw Headers Hex

```
GET /blogs HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Content-Length: 0
Cache-Control: no-cache
Authorization:
eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJlc2VyIiwiaXVkaWVuY2UiOiJ3ZW
IiLCJpc2VhdGVkIjoxNDc0Mzg2NDE1MTIyLCJleHAiOiJ0e0ZQTEyMTV9
.j3Lb_ZDyJTWYqWi3soN6j0ZUONvjvDK9B1F9Zm7gLDmBTnayaCJb0UK7
G7sBxLm7NZceaGLPVWh7dAozPoZ3A
Content-Type: application/json
```

0 matches

Response

Raw Headers Hex

```
HTTP/1.1 200
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0,
must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Type: application/json; charset=UTF-8
Date: Tue, 20 Sep 2016 15:48:23 GMT
Content-Length: 285

[ {
  "id" : 1,
  "blog" : "Hi",
  "created" : 1474386353103,
  "author" : "dbran"
}, {
  "id" : 2,
  "blog" : "This is a dope blog",
  "created" : 1474386353103,
  "author" : "admin"
}, {
  "id" : 3,
  "blog" : "Another dope blog",
  "created" : 1474386353103,
  "author" : "user"
} ]
```

0 matches

588 bytes | 37 millis

Protect Sensitive Resources

The screenshot displays the Burp Suite Professional interface. The title bar reads "Burp Suite Professional v1.7.04 - Temporary Project - licensed to Independent Security Evaluators [9 user license]". The main menu includes "Burp", "Intruder", "Repeater", "Window", and "Help". Below the menu is a toolbar with various options: "Project options", "User options", "Alerts", "Additional Scanner Checks", "Authorize", "CSRF", "Heartbleed", "SessionAuth", "Target", "Proxy", "Spider", "Scanner", "Intruder", "Repeater", "Sequencer", "Decoder", "Comparer", and "Extender".

The interface shows a single tab with the target "http://localhost:8080". The "Request" pane on the left displays a DELETE request to "/blogs/2" with headers including "Host: localhost:8080", "Connection: keep-alive", "Content-Length: 0", "Cache-Control: no-cache", "Authorization: eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJlc2VyIiwiaXVkaWVuY2UiOiJ3ZW...". The "Response" pane on the right shows an HTTP 200 response with headers like "X-Content-Type-Options: nosniff", "X-XSS-Protection: 1; mode=block", "Cache-Control: no-cache, no-store, max-age=0, must-revalidate", "Pragma: no-cache", "Expires: 0", "X-Frame-Options: DENY", and "Content-Type: application/json; charset=UTF-8". The response body is a JSON object: {"id": 2, "blog": "This is a dope blog", "created": 1474386353103, "author": "admin"}.

At the bottom, there are search bars for both the request and response, each showing "0 matches". The status bar at the bottom left says "Ready" and the bottom right shows "401 bytes | 16 millis".

Protect Sensitive Resources

What the code looks like...

```
@RequestMapping(path = "/blogs/{id}", method = RequestMethod.DELETE)  
public Blog removeBlog(@PathVariable("id") long id, HttpServletRequest request) {  
  
    blog = blogs.get(id);  
    return blogs.remove(id);  
}
```

Access controls are not implemented!!

Protect Sensitive Resources

What the code should look like...

```
@RequestMapping(path = "/blogs/{id}", method = RequestMethod.DELETE)
public ResponseEntity<?> removeBlog(@PathVariable("id") long id, HttpServletRequest request) {

    String token = request.getHeader(tokenHeader);
    String username = jwtTokenUtil.getUsernameFromToken(token);
    JwtUser user = (JwtUser) userDetailsService.loadUserByUsername(username);
    Collection<? extends GrantedAuthority> authorities = user.getAuthorities();

    blog = blogs.get(id);

    if (blog.getAuthor().equalsIgnoreCase(username) ||
        authorities.contains(new SimpleGrantedAuthority("ROLE_ADMIN"))){

        blogs.remove(id);
        return new ResponseEntity<> ("Blog was successfully deleted.", HttpStatus.OK);
    }

    else
        return new ResponseEntity<>("You are not authorized to perform this action.",
            HttpStatus.UNAUTHORIZED);
}
```


Protect Sensitive Resources

The screenshot displays the Burp Suite Professional interface. The title bar reads "Burp Suite Professional v1.7.04 - Temporary Project - licensed to Independent Security Evaluators [9 user license]". The menu bar includes "Burp", "Intruder", "Repeater", "Window", and "Help". The toolbar contains various tool categories: "Project options", "User options", "Alerts", "Additional Scanner Checks", "Authorize", "CSRF", "Heartbleed", "SessionAuth", "Target", "Proxy", "Spider", "Scanner", "Intruder", "Repeater", "Sequencer", "Decoder", "Comparer", and "Extender".

The main window shows a "Request" and "Response" view. The "Request" tab is active, showing a DELETE request to `/blogs/2` on `localhost:8080`. The request headers include `Host: localhost:8080`, `Connection: keep-alive`, `Content-Length: 0`, `Cache-Control: no-cache`, and `Authorization: eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJlc2VyIiwiaWF0Ij0iJ3ZWIiLCJjcmVhdGVkIjoxNDc0Mzg2NDE1MTIyLCJleHAiOiJ0e0NzQ5OTYyMTV9.j3Lb_ZDyJTWYqWi3soN6j0ZU0NvjvDK9B1F9Zm7gLdMBTnayuaCJb0UK7G7sBxLm7NZceaGLPVWh7dAozPoZ3A`. The content type is `application/json`.

The "Response" tab is active, showing a 401 Unauthorized response. The response headers include `HTTP/1.1 401`, `X-Content-Type-Options: nosniff`, `X-XSS-Protection: 1; mode=block`, `Cache-Control: no-cache, no-store, max-age=0, must-revalidate`, `Pragma: no-cache`, `Expires: 0`, `X-Frame-Options: DENY`, `Content-Type: text/plain; charset=UTF-8`, `Content-Length: 46`, and `Date: Tue, 20 Sep 2016 17:34:09 GMT`. The body of the response is `You are not authorized to perform this action.`

At the bottom of the interface, there are search bars for both the request and response, each showing "0 matches". The response size is indicated as "342 bytes | 1,397 millis".

Input Validation

- **Validate Incoming Content-Types**
 - Server should never assume the Content-type
 - Content-Type should be checked against the data
 - 400 level status code should be returned if header contains invalid types

Input Validation

Restricting content types

```
@RequestMapping(path = "/blogs", consumes= {"application/json"}, method = RequestMethod.POST)  
public Blog addBlog(@RequestBody Blog blog, HttpServletRequest request){  
  
    String token = request.getHeader(tokenHeader);  
    String username = jwtTokenUtil.getUsernameFromToken(token);  
  
    blog.setId(blogs.size() + 1);  
    blog.setAuthor(username);  
    blog.setBlog(blog.getBlog());  
    blog.getCreated();  
    blogs.put(blog.getId(), blog);  
    return blog;  
}
```

Spring

Input Validation

Restricting content types

```
@POST
@Produces(MediaType.APPLICATION_JSON)
@Consumes(MediaType.APPLICATION_JSON)
public Blog addBlog(Blog blog){

    return blogService.addBlog(blog);
}
```

Jersey

Input Validation

The screenshot displays the Burp Suite Professional v1.7.04 interface. The title bar indicates it is a temporary project licensed to Independent Security Evaluators. The main menu includes Burp, Intruder, Repeater, Window, and Help. A toolbar contains various tools like Extender, Project options, User options, Alerts, Additional Scanner Checks, Authorize, CSRF, Heartbleed, SessionAuth, Target, Proxy, Spider, Scanner, Intruder, Repeater, Sequencer, Decoder, and Comparer. The Repeater tool is active, showing a single request.

Request

Raw Params Headers Hex XML

```
POST /blogs HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Content-Length: 93
Cache-Control: no-cache
Content-Type: text/xml
Authorization:
eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJlc2VyIiwiaWF0IjoiJ3ZWIiLCJpcmlhIjoiVkljoxNDC0Mzg2NDE1MTIyLCJleHAiOiJlE0NzQ5OTEyMTV9.j3Lb_ZDyJTwYqWi3soN6j0ZU0NvjvDK9B1F9Zm7gLDmBTnayaCJb0UK7G7sBxLm7NZceaGLPVWh7dAozPoZ3A
```

<?xml version="1.0" encoding="utf8" ?>
<blogs>
<blog>Interesting blog</blog>
</blogs>

0 matches

Done

Response

Raw Headers Hex

```
HTTP/1.1 415
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Type: application/json; charset=UTF-8
Date: Tue, 20 Sep 2016 23:16:53 GMT
Content-Length: 243
```

```
{
  "timestamp" : 1474413413282,
  "status" : 415,
  "error" : "Unsupported Media Type",
  "exception" :
  "org.springframework.web.HttpMediaTypeNotSupportedException",
  "message" : "Content type 'text/xml' not supported",
  "path" : "/blogs"
}
```

0 matches

546 bytes | 13 millis

Input Validation

Target: http://localhost:8080

Request

Raw Params Headers Hex XML

```
POST /blogs HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Content-Length: 93
Cache-Control: no-cache
Content-Type: application/json
Authorization:
eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJlc2VyTiIiwiaWF0IjoiJ3ZWIiLCJmVhdG
VkIjoxNDc0Mzg2NDE1MTIyLCJleHAiOiJ0e0NzQ5OTEyMTV9.j3Lb_ZDyJTWYqWi3soN6j0ZU
ONvjvDK9B1F9Zm7gLdMBTnayuaCJb0UK7G7sBxLm7NzceaGLPvWh7dAozPoZ3A

<?xml version="1.0" encoding="utf8" ?>
<blogs>
<blog>Interesting blog</blog>
</blogs>
```

0 matches

Done

Response

Raw Headers Hex

```
HTTP/1.1 400
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Type: application/json; charset=UTF-8
Date: Tue, 20 Sep 2016 23:13:31 GMT
Connection: close
Content-Length: 679

{
  "timestamp" : 1474413211497,
  "status" : 400,
  "error" : "Bad Request",
  "exception" :
  "org.springframework.http.converter.HttpMessageNotReadableExcept
ion",
  "message" : "Could not read document: Unexpected character
('<' (code 60)): expected a valid value (number, String, array,
object, 'true', 'false' or 'null')\n at [Source:
java.io.PushbackInputStream@6351d18e; line: 1, column: 2];
nested exception is
com.fasterxml.jackson.core.JsonParseException: Unexpected
character ('<' (code 60)): expected a valid value (number,
String, array, object, 'true', 'false' or 'null')\n at [Source:
java.io.PushbackInputStream@6351d18e; line: 1, column: 2]",
  "path" : "/blogs"
}
```

0 matches

1,001 bytes | 47 millis

Output Encoding

- Security headers should be sent within all responses
 - Content-Type
 - Should contain correct Content-Type
 - X-Content-Type-Options: nosniff
 - Ensures browsers wont attempt to detect a different Content-Type

QUESTIONS??

- Blog API
 - <https://github.com/dbran9/jwt-spring-security>
 - Based partially off
 - <https://github.com/szerhusenBC/jwt-spring-security-demo>
- Presentation
 - www.securityevaluators.com/knowledge/presentations

Send questions to dbranch@securityevaluators.com