

The State of Securing RESTful APIs with Spring

By Rob Winch

@rob_winch

**TOUCHING WIRES CAUSES
INSTANT DEATH**



\$200 FINE



• Newcastle Tramway Authority •

Authentication

Naïve approach...

https://api.example.com?
username=rob&password=secret

Futurama



“ Come on Bender. It's up to you to make your own decisions in life. That's what's separates people and robots from animals .. and animal robots!

Fry



Search for people, jobs, companies, and more...



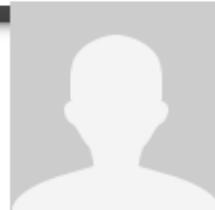
Advanced

3

Skills & Endorsements

Top Skills

1 REST



Mom 1st

Rob's Mom

[Send a message](#)

[View Profile](#)



RFC-7231 Sensitive Information

“

Authors of services ought to avoid GET-based forms for the submission of sensitive data ...

- RFC-7231: Section 9.4

<http://tools.ietf.org/html/rfc7231#section-9.4>

Basic Authentication

HTTP/1.1 401 Access Denied

WWW-Authenticate: Basic realm="Rest"

Content-Length: 0

Basic Authentication

```
GET /messages/100 HTTP/1.1
```

```
Authorization: Basic cm9iOnNlY3JldA==
```

Digest Authentication

HTTP/1.0 401 Unauthorized

WWW-Authenticate: Digest realm="rest@example.com",
qop="auth,auth-int",
nonce="dcd98b7102dd2f0e8b11d0f600bf0c093",
opaque="5ccc069c403ebaf9f0171e9517f40e41"
Content-Length: 0

Digest Authentication

GET /messages/ HTTP/1.0

Authorization: Digest username="rob",
realm="rest@example.com",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="/messages/",
qop=auth,
nc=00000001,
cnonce="0a4f113b",
response="460b693843cc6d2c3b9bde8ec1eeef505",
opaque="5ccc069c403ebaf9f0171e9517f40e41"

Digest Authentication

```
HA1 = MD5( "rob:rest@example.com:secret" )  
= 8ff99f404047cfbf7a5973437dd9453b
```

```
HA2 = MD5( "GET:/messages/" )  
= b3b2c648e81657249f8e940c9aa7a121
```

```
Response = MD5( "8ff99f404047cfbf7a5973437dd9453b:\\  
dcd98b7102dd2f0e8b11d0f600fb0c093:\\  
00000001:0a4f113b:auth:\\  
b3b2c648e81657249f8e940c9aa7a121" )  
= 460b693843cc6d2c3b9bde8ec1eef505
```

Transport Layer Security (TLS)

- Confidentiality
- Integrity



goto fail;



Checking TLS

<https://www.ssllabs.com/ssltest/>

<https://shaaaaaaaaaaaaaa.com/>

TLS Performance

- Computational overhead
- Latency overhead
- Cache

“

On our production frontend machines, **SSL/TLS accounts for less than 1% of the CPU load**, less than 10 KB of memory per connection and less than 2% of network overhead.

- [Adam Langley, Google](#)

<https://www.imperialviolet.org/2010/06/25/overclocking-ssl.html>

“

We have found that modern software-based TLS implementations running on **commodity CPUs are fast enough to handle heavy HTTPS traffic load** without needing to resort to dedicated cryptographic hardware.

- Doug Beaver, Facebook

<http://lists.w3.org/Archives/Public/ietf-http-wg/2012JulSep/0251.html>

“

HTTP keepalives and session resumption mean that most requests do not require a full handshake, so **handshake operations do not dominate our CPU usage.**

- Jacob Hoffman-Andrews, Twitter

<https://blog.twitter.com/2013/forward-secrecy-at-twitter-0>

TLS Optimize

- TLS Resumption
- Latency
- Online Certificate Status Protocol (OCSP)
- Cloudflare

Optimizing TLS

Is TLS Fast Yet.com

HTTP Basic over HTTPS?

oclHashcat

Hash Type	Speed
SHA1	42.408 Bh/s
SHA256	16.904 Bh/s
SHA512	5.2 Bh/s

Ubuntu 14.04, 64 bit
ForceWare 346.29
X NVidia Titan X

Introduce Session

username=winch&name=Rob+Winch

Encrypting the Session

```
Base64(IV,  
        aes_cbc(k,IV,plainText))
```

- **k** – a secret key only known to server
- **aes_cbc** – encrypts the plainText using AES/CBC with the provided IV
- **plainText** – format of username=winch&name=Rob+Winch



Your handwriting is atrocious, not
encrypted

Introduce Session

Can change properly encrypted value below:

`username=winch&name=Rob+Winch`

To have the following Plaintext

`username=admin&name=Rob+Winch`

<https://github.com/rwinch/encryption-not-signing>



Tony Arcieri

@bascule



Follow

JOSE/JWT considered harmful and
dangerous (ask me why!)

2:03 PM - 27 Jul 2015

<https://twitter.com/bascule/status/62577358144877728>



Thomas H. Ptacek

@tqbf

+ Follow

Thinking about securing an API with JWT?
First, punch yourself in the face. Then: just
use a 256 bit random token, and a database.

10:54 AM - 28 May 2015

<https://twitter.com/tqbf/status/603982599459971072>

JWT Header

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

HOW TO INSULT A DEVELOPER

IT'S NOT
RESTFUL



“

... each request from client to server must contain all of the information necessary to understand the request, and **cannot take advantage of any stored context on the server.**

- Roy Fielding, [Architectural Styles and the Design of Network-based Software Architectures](#)

<http://bit.ly/1NpKeJz>

Representational STATE transfer

“

... session state can be transferred by the server to another service such as a database to maintain a persistent state for a period and allow authentication

- [Wikipedia](#)

<http://bit.ly/1R12M0t>



Code Slide

```
public interface HttpSession {
```

```
    ...
```

```
}
```

GET /messages/ **HTTP/1.1**

Host: api.example.com

Cookie: JSESSIONID=AS348AF929FK219CKA9FK3B79870H; HttpOnly; secure;

Accept: application/json

Customizing the Cookie

```
<session-config>
    <cookie-config>
        <name>SESSION</name>
    </cookie-config>
</session-config>
```

Spring Session

```
@Configuration  
@EnableRedisHttpSession  
public class Config {  
  
    @Bean  
    public JedisConnectionFactory connectionFactory() {  
        return new JedisConnectionFactory();  
    }  
}
```

Spring Session

```
public class Initializer extends  
    AbstractHttpSessionApplicationInitializer {  
  
    public Initializer() {  
        super(Config.class);  
    }  
}
```

Spring Session

```
public class Initializer extends  
    AbstractHttpSessionApplicationInitializer {  
  
    public Initializer() {  
        super(Config.class);  
    }  
}
```

Spring Session

```
<filter>
  <filter-name>
    springSessionRepositoryFilter
  </filter-name>
  <filter-class>
    org.springframework.web.filter.DelegatingFilterProxy
  </filter-class>
</filter>

<filter-mapping>
  <filter-name>springSessionRepositoryFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

GET /messages/ HTTP/1.1

Host: api.example.com

x-auth-token: AS348AF929FK219CKA9FK3B79870H

Accept: application/json

SessionRepositoryFilter

```
public void doFilter(ServletRequest req,  
                     ServletResponse resp,  
                     FilterChain chain {  
  
    ServletRequest request =  
        new SessionRepositoryRequestWrapper(req);  
    ...  
    chain.doFilter(request, response);  
}
```

SessionRepositoryRequestWrapper

```
public HttpSession getSession() {  
    // return custom HttpSession  
}
```

OAuth 2.0?

- When working within a sandbox
- Limiting liability



Authorization

```
http
    .authorizeRequests()
        .antMatchers("/public/**").permitAll()
        .antMatchers("/admin/**").hasRole("ADMIN")
        .anyRequest().authenticated()
```

Authorization

```
@PostAuthorize("returnObject?.to?.id == principal.id")
Message findOne(Long id);
```

Authorization

```
@PreAuthorize("#message?.from?.id == principal.id")
<S extends Message> S save(Message message);
```

Permissions

```
@PostAuthorize("hasPermission(returnObject, 'read')")  
Message findOne(Long id);
```

Permissions

```
@PreAuthorize("hasPermission(#message, 'write')")  
<S extends Message> S save(Message message);
```

```
public interface PermissionEvaluator ... {  
    boolean hasPermission(Authentication authentication,  
                          Object targetDomainObject,  
                          Object permission);  
    boolean hasPermission(Authentication authentication,  
                          Serializable targetId,  
                          String targetType,  
                          Object permission);  
}
```

Queries?

```
@Query("select m from Message m where m.to.id = ?  
#{principal.id}")  
Iterable<Message> inbox();
```

Queries?

```
@Query("select m from Message m where m.to.id = ?  
#{principal.id}")  
Page<Message> inbox(Pageable pageable);
```

Future Work?

```
@EnableAclSecurity  
public interface SecuredMessageRepository  
    extends MessageRepository {}
```

// Vote for it! [DATACMNS-293 SEC-2409](#)

// <https://jira.spring.io/browse/DATACMNS-293>
// <https://jira.spring.io/browse/SEC-2409>

CSRF Protection

Win Money

CSRF Protection

POST /messages/100 HTTP/1.1

Host: localhost:8080

Content-Type: application/x-www-form-urlencoded

Cookie: SESSION=7bb7792b-b640-45a3-bec8-bacb3ca95cbb

_method=delete

CSRF Protection

POST /messages/100 HTTP/1.1

Host: localhost:8080

Content-Type: application/x-www-form-urlencoded

Cookie: SESSION=7bb7792b-b640-45a3-bec8-bacb3ca95cbb

X-CSRF-TOKEN: 322c2de6-00ab-4bb3-8a97-ab3c7291d4ad

_method=delete

CSRF Protection

“ When do I use CSRF protection?

CSRF Protection

“ ... but my application uses JSON

CSRF Protection

```
<form ... method="post" enctype="text/plain">
  <input type='hidden'
    name='{"summary": "Hi ", ... "ignore_me": "'"
value='test"'>
</form>
```

CSRF Protection

```
{  
  "summary": "Hi",  
  "message": "New Message",  
  "to": "luke@example.com",  
  "ignore_me": "=test"  
}
```

CSRF Protection

“ ... but my application is stateless

CSRF Protection

POST /sample/100 HTTP/1.1

Host: localhost:8080

Content-Type: application/x-www-form-urlencoded

Authorization: Basic cm9iQGV4YW1wbGUuY29tOnBhc3N3b3Jk

_method=delete

CSRF Protection

“ ...and I use a custom header for authentication and ignore cookies

CSRF Protection

- Use proper HTTP Verbs
- Configure CSRF Protection
- Include the CSRF Token

Including the CSRF Token

```
@RequestMapping("/csrf")
public CsrfToken csrf(CsrfToken token) {
    return token;
}
```

Security HTTP Response Headers

Win Money

Security HTTP Response Headers

WinMoney
Delete

Delete

Delete

Security HTTP Response Headers

HTTP/1.1 200 OK

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

Strict-Transport-Security: max-age=31536000 ; includeSubDomains

Learn More. Stay Connected.



- Use TLS
- Authentication Should Have State
- Use Proper Authorization
- Use a Framework Because Individuals Cannot Provide Good Security

Twitter: [@rob_winch](https://twitter.com/rob_winch)

YouTube: [spring.io/video](https://www.youtube.com/springio)

LinkedIn: [spring.io/linkedin](https://www.linkedin.com/company/spring-framework-project/)

Google Plus: [spring.io/gplus](https://plus.google.com/+SpringFramework)