Secure Coding Practices in Java: Challenges and Vulnerabilities

Na Meng, Stefan Nagy, Daphne Yao, Wenjie Zhuang, Gustavo Arango Argoty
Presented by: Md Mahir Asef Kabir

Problem Statement

- Java platform and third-party libraries provide security features
- Misusing the features cost time and effort or cause vulnerabilities
  - Bypassing certificate validation, Using Broken Hashing algorithm, Disabling Cross Site Resource Forgery policy etc.
- Prior research focused on misuse of cryptography and SSL APIs
- This paper investigated the common concerns, programming challenges and security vulnerabilities

Research Questions

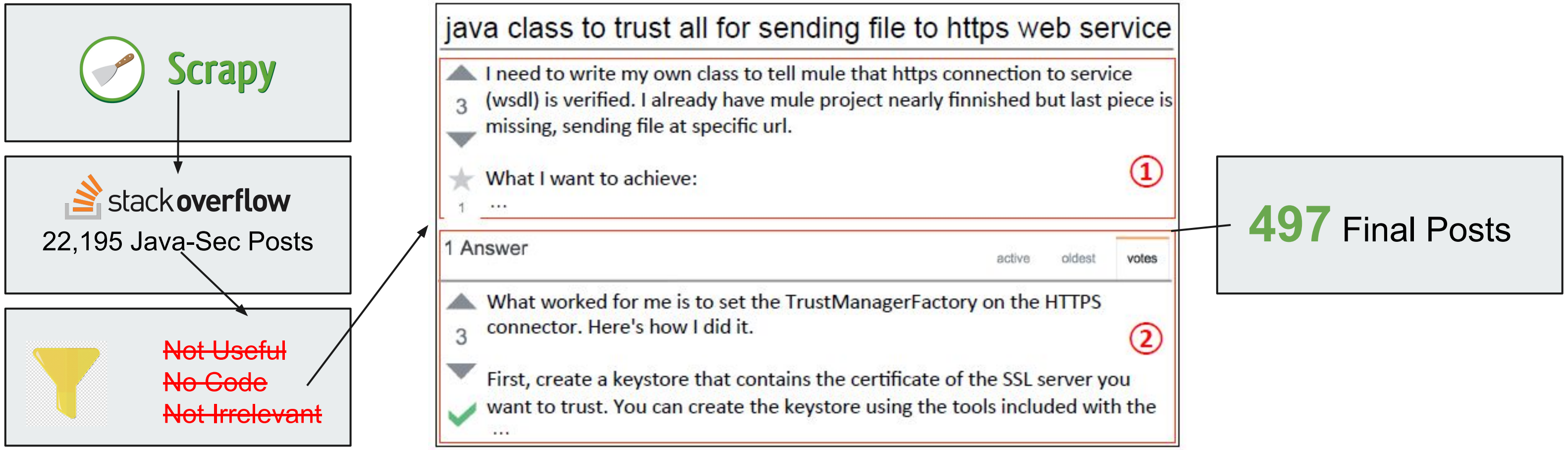
- What are the common concerns on Java secure coding?
    - What are the most popular asked about security features?
    - What are the Hard-to-implement security defenses in practice?
- What are the common programming challenges?
    - Why developers could not write secure code?
- What are the common security vulnerabilities?

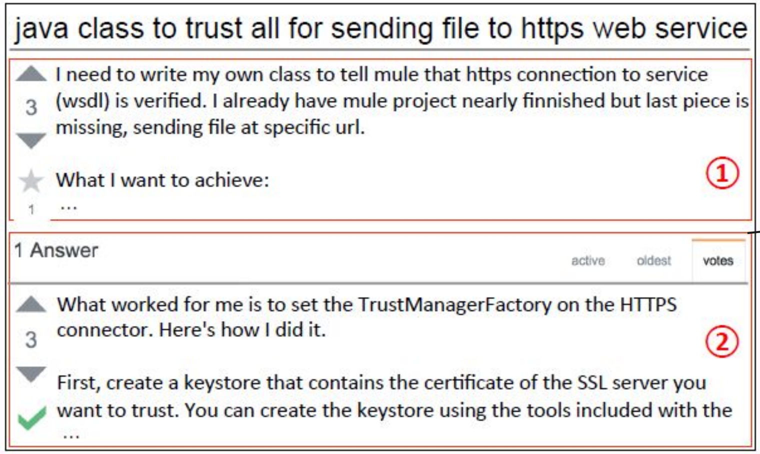
Background

Stackoverflow posts covered 3 main perspectives

- Java Platform Security
  - Areas - Cryptography, Access Control & Secure Communication
- Java EE Security
  - Two ways to implement - Declarative Security & Programmatic Security
- Third Party Frameworks
  - Spring Security

Approach




java class to trust all for sending file to https web service

3

I need to write my own class to tell mule that https connection to service (wsdl) is verified. I already have mule project nearly finnished but last piece is missing, sending file at specific url.

What I want to achieve:

1

...

①

1 Answer

active oldest votes

3

What worked for me is to set the TrustManagerFactory on the HTTPS connector. Here's how I did it.

②

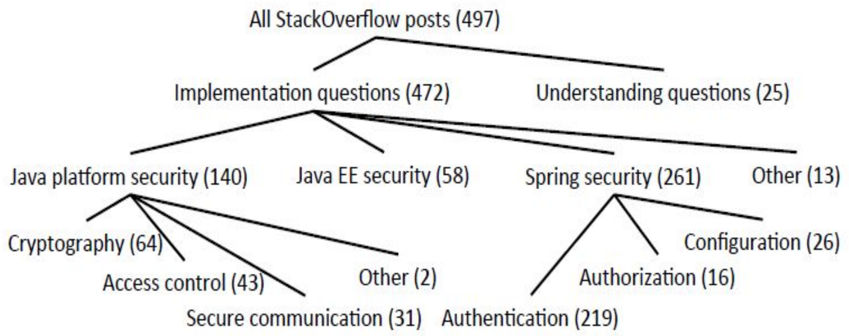
First, create a keystore that contains the certificate of the SSL server you want to trust. You can create the keystore using the tools included with the

...

[*] https://medium.com/@chetaniam/using-scrapy-to-create-a-generic-and-scalable-crawling-framework-83d36732181
[**] https://stackoverflow.design/brand/logo/ [***] https://www.hiclipart.com/free-transparent-background-png-clipart-jcsvb
[****] Figure 1 of paper: [A highly viewed post asking about workarounds to bypass key checking and allow all host names for HTTPS]

Evaluation

Classification hierarchy among 497 posts [*]



[*] Figure 2 of paper: Taxonomy of StackOverflow posts

Evaluation (Contd.)

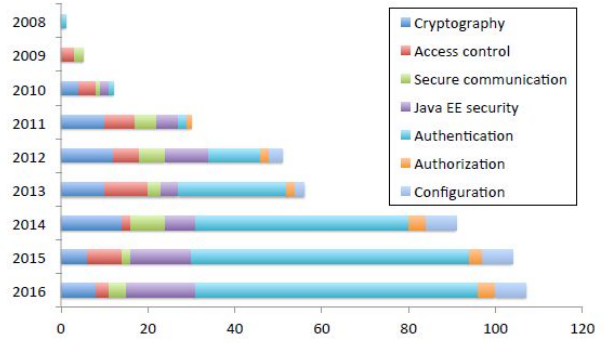
- Cryptography, Access Control, Server Communication [Java Platform Security (**30%**)]

- Authentication, Authorization, Configuration [Spring Security (**50%**)]

- Java EE Security (**12%**)

- **More questions on Java Enterprise Applications**

Post distribution after 3rd level classification [*]

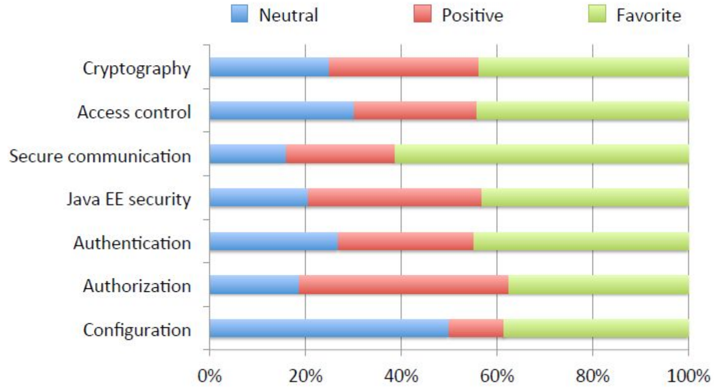


[*] Figure 3 of paper: The post distribution during 2008-2016

Evaluation (Contd.)

- Clustered posts based on developers' attitude towards the question
- Defined 3 types of sentiments - **neutral**, **positive**, **favorite**
- Secure Communications related questions are most favorite (61%)
- Developers focus more on security implementation instead of environment settings

Developers' sentiment on questions

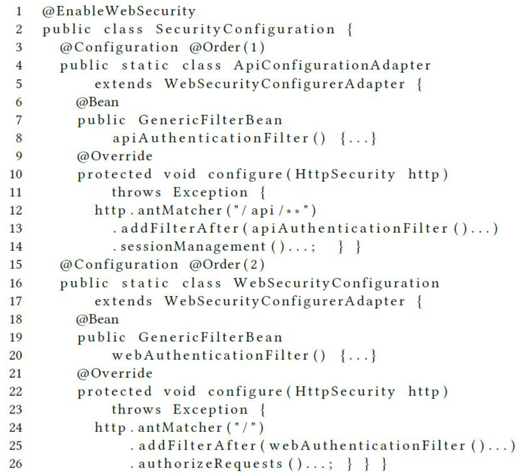


[*] Figure 3 of paper: The post distribution among developers' sentiment towards the security features: neutral, positive, and favorite

Common Programming Challenges

Authentication

- Variations in way to integrate Spring security with different types of applications
- Java and XML-based security configurations hard to implement correctly
- Conversion from XML-based to Java-based security is tedious & error-prone



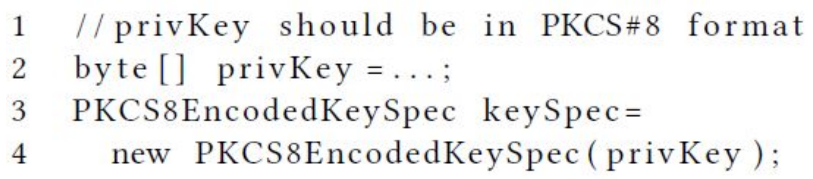
```
@EnableWebSecurity
public class SecurityConfiguration {
  @Configuration @Order(1)
  public static class ApiConfigurationAdapter
      extends WebSecurityConfigurerAdapter {
    @Bean
    public GenericFilterBean
        apiAuthenticationFilter() {...}
    @Override
    protected void configure(HttpSecurity http)
        throws Exception {
      http.antMatcher("/api/**")
        .addFilterAfter(apiAuthenticationFilter()...)
        .sessionManagement()...;   } }
  @Configuration @Order(2)
  public static class WebSecurityConfiguration
      extends WebSecurityConfigurerAdapter {
    @Bean
    public GenericFilterBean
        webAuthenticationFilter() {...}
    @Override
    protected void configure(HttpSecurity http)
        throws Exception {
      http.antMatcher("/")
          .addFilterAfter(webAuthenticationFilter()...)
          .authorizeRequests()...; } } }
```

[*] Listing 1 of paper: An exemplar implementation working unexpectedly in Spring Boot applications

Common Programming Challenges (Contd.)

Cryptography

- Error message not providing sufficient hints - Getting same exceptions for missing steps
- Difficult to implement security with multiple programming languages
- Implicit constraint on API usage causing confusion



```
//privKey should be in PKCS#8 format
byte[] privKey =...;
PKCS8EncodedKeySpec keySpec=
   new PKCS8EncodedKeySpec(privKey);
```

[*] Listing 3 of paper: Consistency between the key format and keyspec

Common Programming Challenges (Contd.)

Java EE Security

- Developers misunderstand annotations
- Possible to use incorrect conflicting annotations
- No tool for preventing

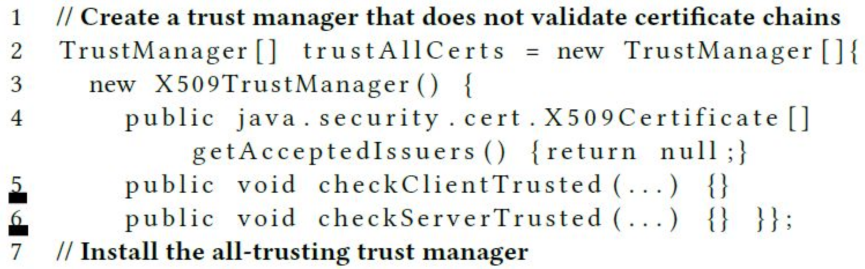
Access Control

- Effect of access control varies with the program context
- Effect of access control varies with the execution environment

Common Programming Challenges (Contd.)

Secure Communications

- Unable to find valid server certificate
- Accepted answers suggesting to disable SSL verification process

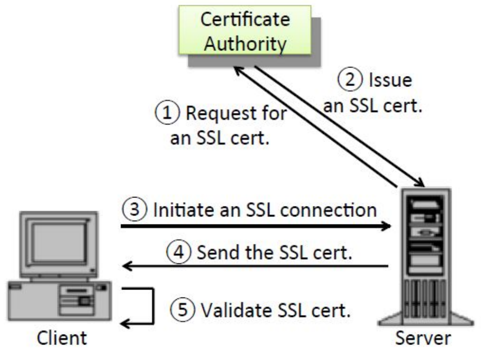


```
// Create a trust manager that does not validate certificate chains
TrustManager[] trustAllCerts = new TrustManager[]{
  new X509TrustManager() {
    public java.security.cert.X509Certificate[]
        getAcceptedIssuers() {return null;}
    public void checkClientTrusted(...) {}
    public void checkServerTrusted(...) {} }};
// Install the all-trusting trust manager
```

[*] From Listing 4 of paper: A typical implementation to disable SSL certificate validation

Common Problems from Security Perspectives

- Disabling Cross-site request forgery protection
- SSL/TLS
  - Trusting all SSL certificates
  - Unaware of the best usages
- Password hashing with MD5 or SHA-1
  - Vulnerable to dictionary attacks



[*] Figure 5 of paper: Simplified overview of creating an SSL connection

Related Works

Analyzing Security Vulnerabilities

- Identifying Java features whose misuse can compromise security (e.g. - Using reflection to access normally inaccessible fields) [1]
- Examining vulnerabilities from CVE database to find root-cause [2]
- Clustering security related Stackoverflow posts based on Text [3]

Novelty: In-depth Investigation of programming challenges and security vulnerabilities

Related Works (Contd.)

Detecting Security Vulnerabilities

- Detecting violations of 6 well defined Android cryptographic API usage rules [4]
- Manually labelling “secure” or “insecure” to train a classifier to efficiently judge the whole dataset [5]
- Implementing man-in-the-middle attack to reveal vulnerabilities [6]

Novelty: Broader scope (secure coding practice, spring security, poor error message etc.). Usage of Stackoverflow to provide community perspective of secure coding

Related Works (Contd.)

Preventing Security Vulnerabilities

- Creating a security-oriented subset of Java to enforce secure software development (e.g. - Allowing least access privilege by default) [7]
- Implementing library to simplify usage of Cryptography [8]

Recommendations

- For security developers
  - Conduct security testing to verify feature functionality
  - Be cautious when following Stackoverflow accepted answers

- For library designers
  - Design clean and helpful error messages
  - Design simplified APIs with strong defenses implemented by default

- For tool builders
  - Develop automatic tools to diagnose security errors

Conclusion

- Developers do not appear to understand security implications
- Provided evidence showing Spring Security lacks simplicity and proper documentation
- Dynamics among asker and responder influence people’s security choice
    - Insecure answers from high reputed users get accepted
    - Correct answers from low reputed users get ignored by askers

Discussion

- Can we build a tool that can verify if encryption-decryption in different languages are converting correctly?
- What are the pros and cons of doing similar research for NodeJS (the most popular technology in Stack Overflow in 2020)?

References

[1] Fred Long. 2005. Software Vulnerabilities in Java. Technical Report CMU/SEI-2005- TN-044. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=7573

[2] David Lazar, Haogang Chen, Xi Wang, and Nickolai Zeldovich. 2014. Why Does Cryptographic Software Fail?: A Case Study and Open Problems. In Proceedings of 5th Asia-Pacific Workshop on Systems (APSys '14). ACM, New York, NY, USA, Article 7, 7 pages. https://doi.org/10.1145/2637166.2637237

[3] Xin-Li Yang, David Lo, Xin Xia, Zhi-Yuan Wan, and Jian-Ling Sun. 2016. What Security Questions Do Developers Ask? A Large-Scale Study of Stack Overflow Posts. Journal of Computer Science and Technology 31, 5 (01Sep2016),910–924.https://doi.org/10.1007/s11390-016-1672-0

References (Contd.)

[4] Manuel Egele, David Brumley, Yanick Fratantonio, and Christopher Kruegel. 2013. An Empirical Study of Cryptographic Misuse in Android Applications. In Proceedings of the 2013 ACM SIGSAC Conference on Computer &#38; Communications Security (CCS ’13). ACM, New York, NY, USA, 73–84. https://doi.org/10.1145/ 2508859.2516693

[5] Felix Fischer, Konstantin BÂÍottinger, Huang Xiao, Christian Stransky, Yasemin Acar, Michael Backes, and Sascha Fahl. 2017. Stack Overflow Considered Harmful? The Impact of Copy&Paste on Android Application Security. In 38th IEEE Symposium on Security and Privacy (S&P ’17) (2017-05-22).

[6] Martin Georgiev, Subodh Iyengar, Suman Jana, Rishita Anubhai, Dan Boneh, and Vitaly Shmatikov. 2012. The Most Dangerous Code in the World: Validating SSL Certificates in Non-browser Software. In Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS ’12). ACM, New York, NY, USA, 38–49. https://doi.org/10.1145/2382196.2382204

References (Contd.)

[7] Adrian Mettler, David Wagner, and Tyler Close. 2010. Joe-E: A Security-Oriented Subset of Java. In Network and Distributed Systems Symposium. Internet Society. http://www.truststc.org/pubs/652.html

[8] Adrian Mettler, David Wagner, and Tyler Close. 2010. Joe-E: A Security-Oriented Subset of Java. In Network and Distributed Systems Symposium. Internet Society. http://www.truststc.org/pubs/652.html

Thank you