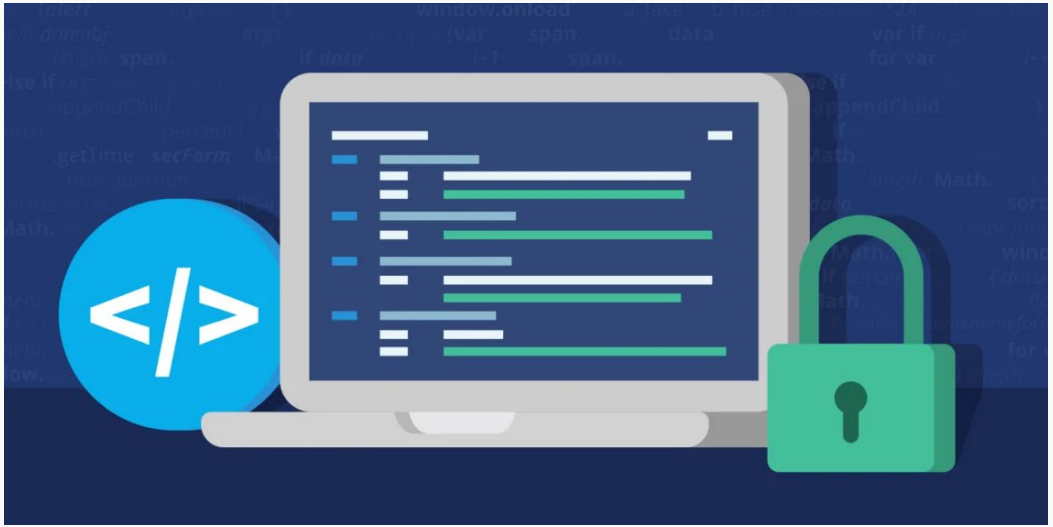
Secure Coding Practices



Gaurav Sood


ETHICAL HACKER
UNITED

Agenda

1. Secure Code - Why?
2. OWASP Top 10
3. Demos
4. Basic Security Practices

The Why?

1. It takes one chink in the armour
2. Results:
   - Loss of system access
   - Information Disclosure
   - Operational Interruption
   - Financial Loss
   - Reputation
   - Lawsuits

OWASP Top 10

- OWASP -> Open Worldwide Application Security Project
- Top 10 -> list of most critical security risks to web applications
- Current List:
  - Broken Access Control
  - Cryptographic Failure
  - Injection
  - Insecure Design
  - Security Misconfiguration
  - Vulnerable Libraries
  - Server Side Request Forgery (SSRF)
  - Data Integrity Failures
  - Authentication Failures
  - Monitoring Failures

Broken Access Control

- Users can do stuff outside of their permission set
- Violation of Least Privilege
- Missing access control policies

Broken Access Control

Impact

- accessing profile of other users
- elevation of privilege
- accessing unauthorized pages

Vulnerable Libraries

Apache Log4j Core » 2.17.0

Implementation for Apache Log4J, a highly configurable logging tool that focuses on performance and low garbage generation. It has a plugin architecture that makes it extensible and supports asynchronous logging based on LMAX Disruptor.

License	Apache 2.0
Categories	Logging Frameworks
Tags	logging log4j apache
Date	Dec 18, 2021
Files	pom (22 KB) jar (1.7 MB) View All
Repositories	Central
Ranking	**#52 in MvnRepository (See Top Artifacts)** **#6 in Logging Frameworks**
Used By	**10,594 artifacts**
Vulnerabilities	**Direct vulnerabilities:** CVE-2021-44832 **Vulnerabilities from dependencies:** CVE-2023-6378 CVE-2022-45868 CVE-2022-42004 View 20 more ...

Injection

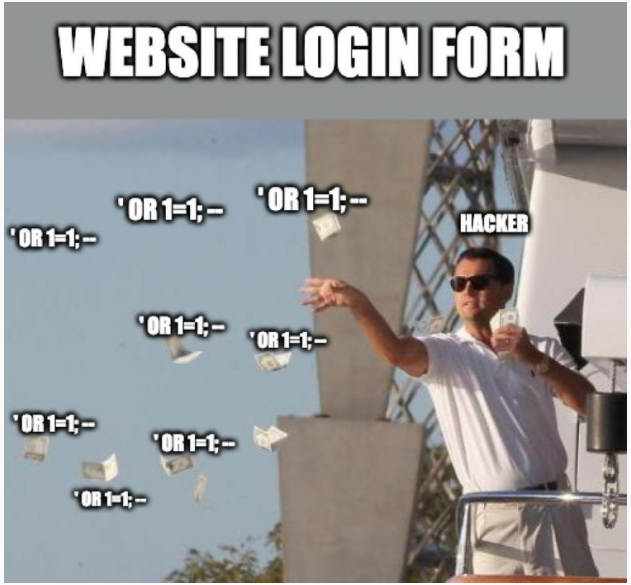
- Untrusted user data sent to application that can lead an attacker to run commands or access unauthorized data
- Common Types:
  - SQL Injection (SQLi)
  - Cross Site Scripting (XSS)
  - Command Injection

Injection

- Impact
  - leaking privileged information
  - remote code execution

SQLi

- Payloads for params that directly deal with the database
- Impact
  - Information disclosure
  - Tamper with existing information



XSS

- Malicious scripts injected into websites
- Impact
  - Session Hijacking
  - Data theft / disclosure
  - Phishing

Command Injection

- OS commands executed through payloads
- Impact
  - System compromise
  - RCE
  - Privilege Escalation

Security Misconfiguration

- Default configuration -> credentials, initial settings
- Unencrypted communication
- Excessive permissions

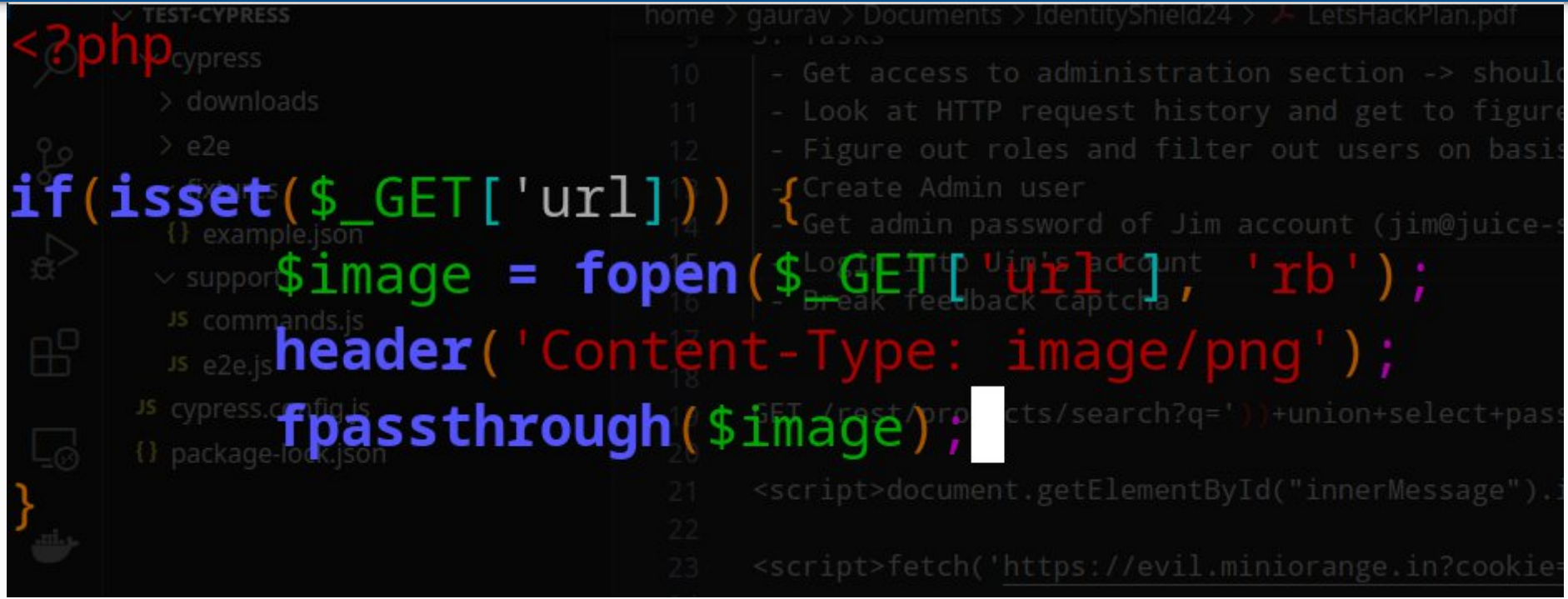
Cryptographic Failures

- Failures / lack of cryptographic controls
- Common Failures:
  - Weak cryptographic ciphers
  - Lack of TLS
  - Unsalted password hashes

Impact

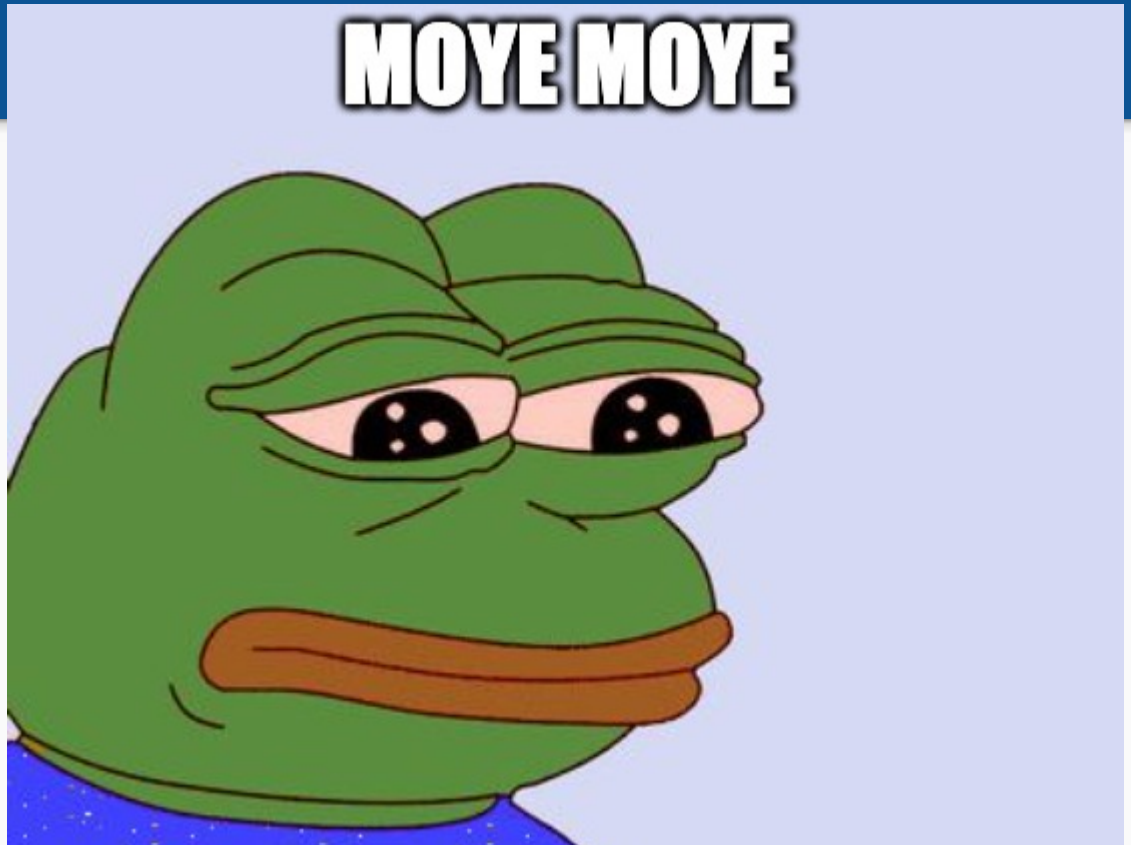
- cracking password hashes through Rainbow Tables
- spying on network traffic

Server Side Request Forgery (SSRF)



Demos

1. Broken Access Control -> Path Traversal
2. Injection -> Cross Site Scripting (XSS)


MOYE MOYE

Solution

Basic Security Practices

1. Validate Input
2. Heed Warnings
3. Securely Architect
4. Keep it Simple
5. Deny By Default
6. Least Privilege
7. Security Testing
8. Defense In Depth

Validate Input

- Accept properly formed data
- Syntactic and semantic
- Regex for preventing SQLi, XSS, Path Traversal
- File uploads

Heed Warnings

- IDEs
- Compilers / Interpreters
- Dependency management (maven, npm)

Securely Architect

- Secure Communication between resources
- Encryption of data at rest
- RBAC
- Secure access to servers
- Audit access and events extensively

Keep It Simple

- Simplicity as key factor in code design
- Easier unit, integration and security tests
- Maintainable

Deny by Default

- Unless explicitly authenticated and authorized

Least Privilege

- Access only resources for which user has access to with minimum privileges
- Scoped APIs
- Role Based Access Control (RBAC)
- Separation of Duties (SoD)

Security Testing

- Static Code Analysis (SAST)
- Dynamic Code Analysis (DAST)
- Penetration Testing

Defense In Depth

- Multiple layers of security
- Authorization at infrastructure and application level
- Sanity checks everywhere
- MFA

Code Review

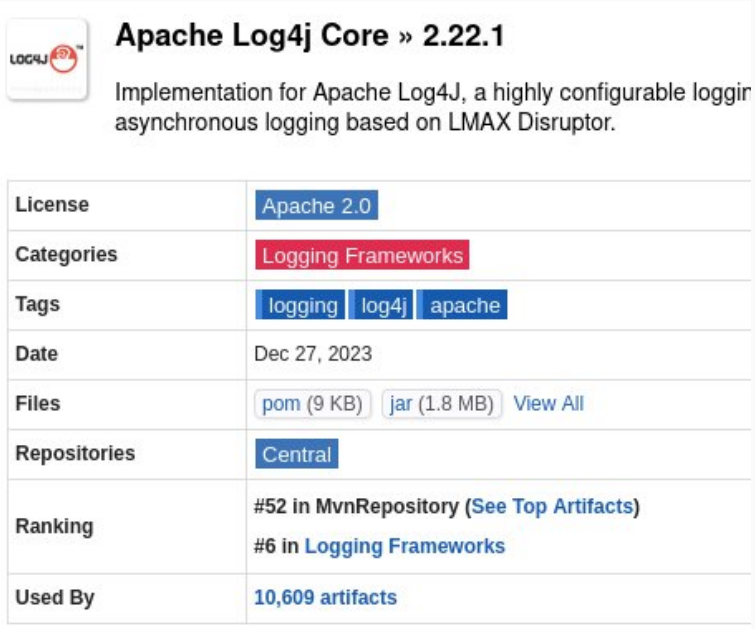
- Extensive code review
- Open Redirects,
- Improper Input Sanitization
- Vulnerable Libraries
- Lower grade encryption / hashing
- Improper integrity checks

Code Review

```
3
4   // 0xRAYAN7 - source code challange (1)
5
6   @PostMapping("/{email}/userroles")
7   public ResponseEntity<?> updateUserRole(Authentication authentication,
8                                           @PathVariable String email, @RequestBody UserRoleUpdateDto roleUpdate) {
9       User authenticatedUser = (User) authentication.getPrincipal();
10      Role newRole = roleUpdate.getNewRole();
11
12      if (authenticatedUser.getEmail().equals(email) || authenticatedUser.hasRole("ADMIN")) {
13          userRoleService.updateUserRole(email, newRole);
14          logger.info("Add User {} to the role {}", email, newRole);
15          return ResponseEntity.ok().build();
16      } else {
17          logger.warn("Unauthorized role update attempt by user {}", authenticatedUser.getEmail());
18          throw new ResponseStatusException(HttpStatus.FORBIDDEN, "You do not have permission to update roles.");
19      }
20  }
21
```

Patch Libraries

- Continuous patching of dependencies
- Static Code Analysis



Apache Log4j Core » 2.22.1

Implementation for Apache Log4J, a highly configurable loggir asynchronous logging based on LMAX Disruptor.

License	Apache 2.0
Categories	Logging Frameworks
Tags	logging log4j apache
Date	Dec 27, 2023
Files	pom (9 KB) jar (1.8 MB) View All
Repositories	Central
Ranking	#52 in MvnRepository (See Top Artifacts) #6 in Logging Frameworks
Used By	10,609 artifacts

Takeaways

- Why secure code and application security is important
- OWASP Top 10 and common vulnerabilities
- Basic Security Practices

Questions?

Feedback


THANK
YOU!