MENTORMATE™
DevCamp

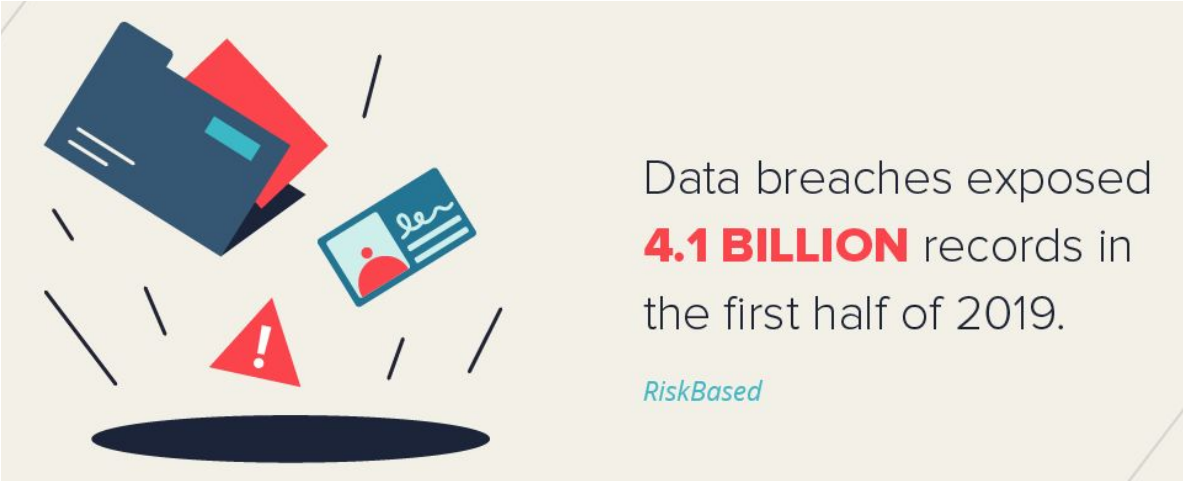
Security

Yulia Tenincheva

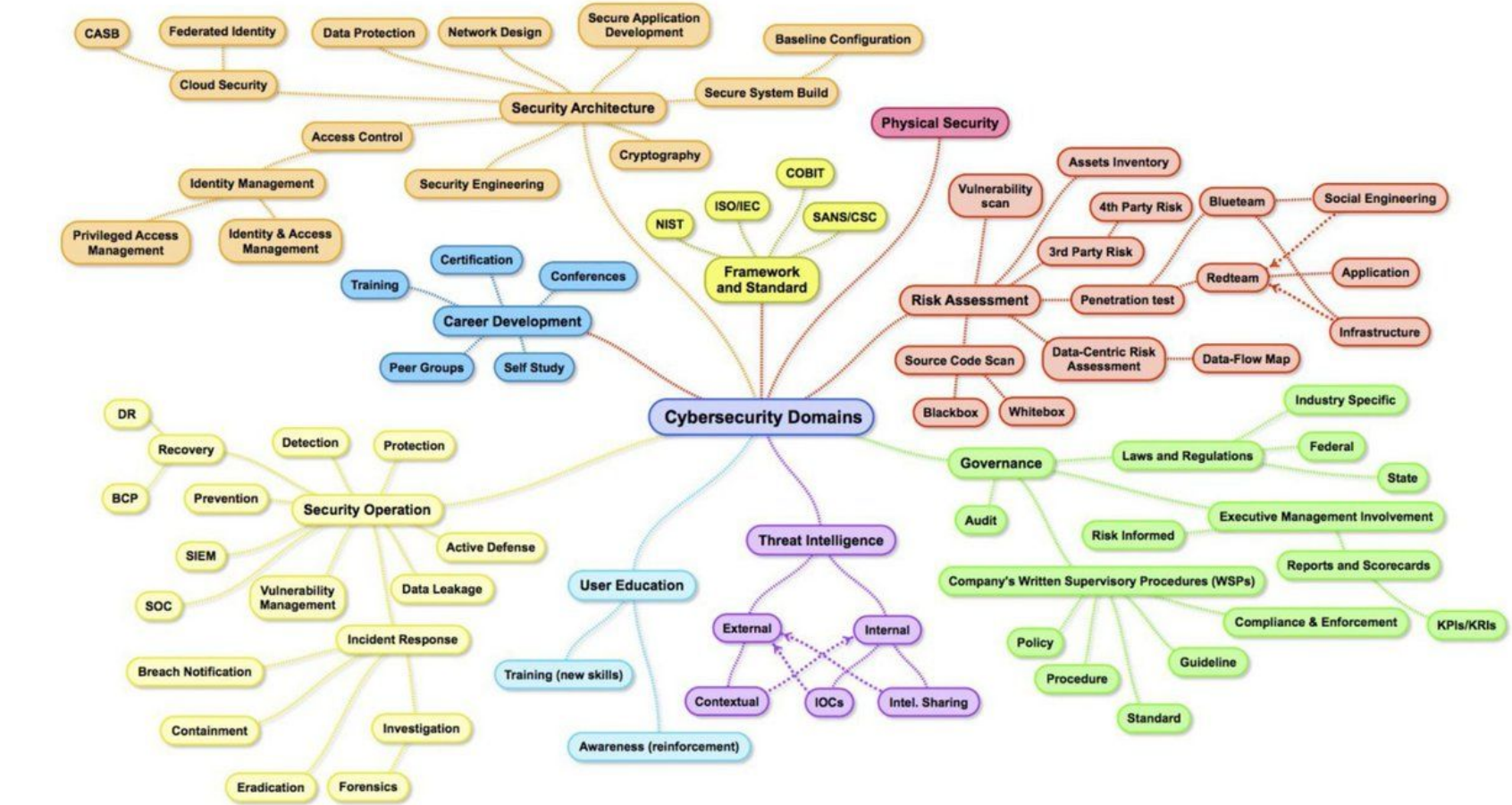
Senior Cloud Engineer, MentorMate

What is the impact of poor security?

- Economic costs
- Reputational costs
- Regulatory costs



Source: Statistics


Cybersecurity Domains
Security Architecture
CASB
Federated Identity
Cloud Security
Data Protection
Network Design
Secure Application Development
Baseline Configuration
Secure System Build
Access Control
Cryptography
Identity Management
Security Engineering
Privileged Access Management
Identity & Access Management
Framework and Standard
COBIT
ISO/IEC
SANS/CSC
NIST
Career Development
Certification
Conferences
Training
Peer Groups
Self Study
Physical Security
Risk Assessment
Vulnerability scan
Assets Inventory
4th Party Risk
3rd Party Risk
Penetration test
Blueteam
Social Engineering
Redteam
Application
Infrastructure
Source Code Scan
Data-Centric Risk Assessment
Data-Flow Map
Blackbox
Whitebox
Governance
Industry Specific
Laws and Regulations
Federal
State
Audit
Executive Management Involvement
Risk Informed
Reports and Scorecards
Company's Written Supervisory Procedures (WSPs)
Compliance & Enforcement
KPIs/KRIs
Policy
Guideline
Procedure
Standard
Threat Intelligence
External
Internal
Contextual
IOCs
Intel. Sharing
User Education
Training (new skills)
Awareness (reinforcement)
Security Operation
DR
Recovery
BCP
Detection
Protection
Prevention
SIEM
Active Defense
SOC
Vulnerability Management
Data Leakage
Incident Response
Breach Notification
Containment
Investigation
Eradication
Forensics

Security is everyone's responsibility!

Today's Agenda

- Security goals and principles
- Web Security
  - Authentication vs Authorization
  - Cookies & Tokens
  - Hashing vs Encryption
  - Security Scanning
  - Secrets management
  - Server security checklist

Security Goals

The CIA triad

- Confidentiality

  Confidentiality measures protect information from unauthorized access and misuse.

- Integrity

  Integrity measures protect information from unauthorized alteration.

- Availability

  Availability measures protect timely and uninterrupted access to the system.

Other general principles

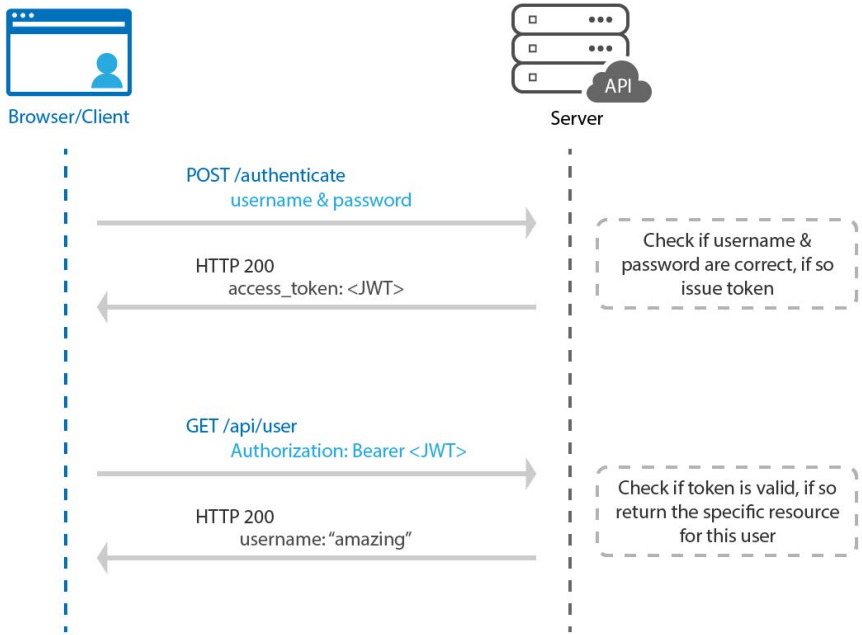
- **Least Privilege**
- **Defense in Depth**
- **KISS** - Keep It Simple and thus Secure
- **Authentication** and **Authorization**
- **Accountability** and **Auditing**
- **Security by design** - principles

Authentication & Authorization

Authentication is the process of verifying who a user is, while Authorization is the process of verifying what they have access to!

Identification, Authentication & Authorization

- Secure user authentication in Node.js
  - Passport.js
- MFA with Node.js
  - Tutorial with Okta in Express.js
- Passwordless
  - Third-party authentication
  - Slack's idea
- Roles



Cookies & Tokens

Cookies & Sessions

- What is a cookie?
  - ES6 Cookies & usage example with Express.js

    small piece of data stored on the user's computer by the web browser while browsing a website.

- What is a Session?
  - Usage examples with Express.js

    A session is a place to store data that you want access to across requests. Each user that visits your website has a unique session.

  - Storing session data in cookies

- Cookies vs sessions vs tokens
  - Explained here and here

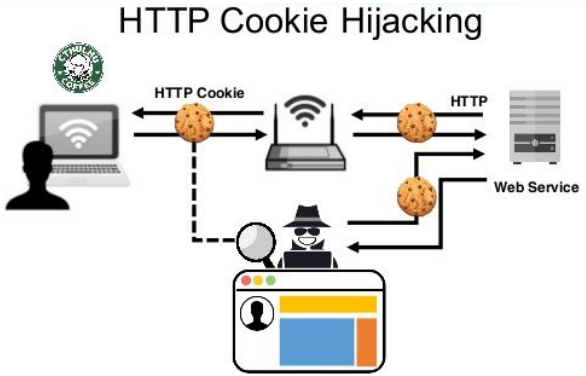
Cookies vs. Sessions

Cookies

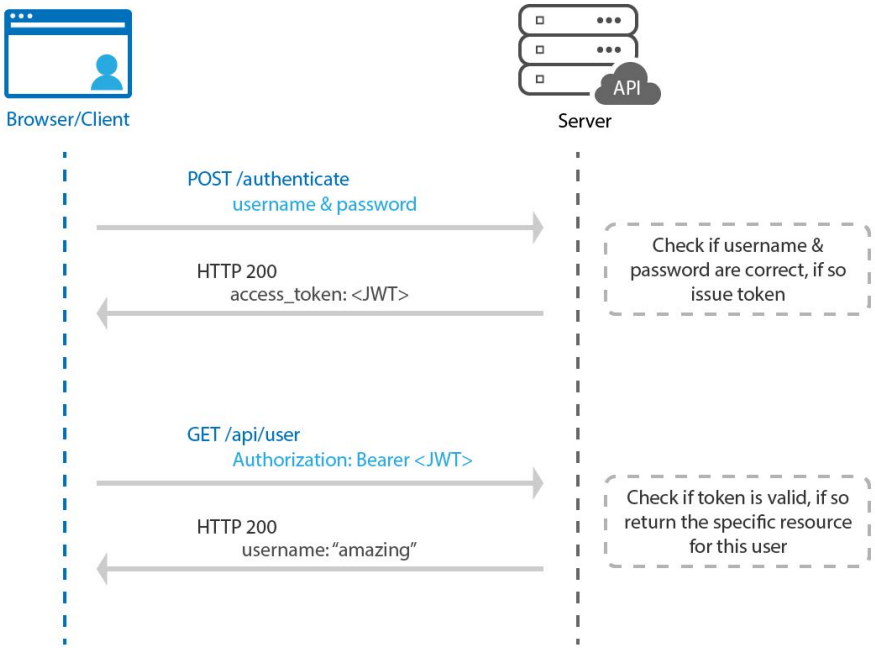
- Cookies are stored on client side
- Cookies can only store strings.
- Cookies can be set to a long lifespan.

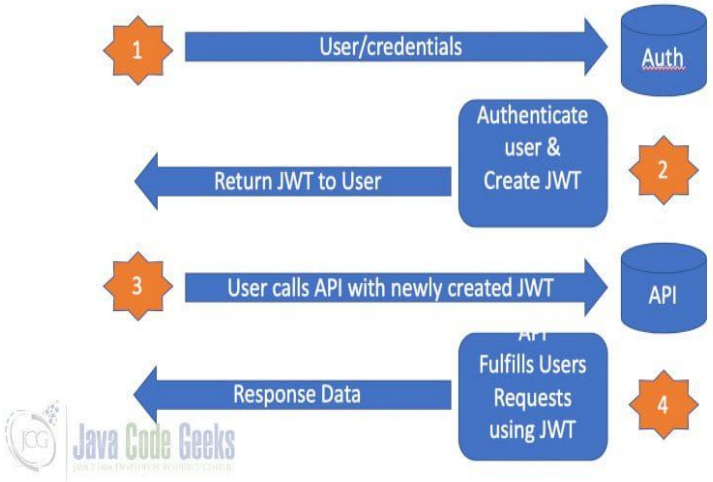
Sessions

- Sessions are stored on server side
- Sessions can store objects.
- When users close their browser, they also lost the session.



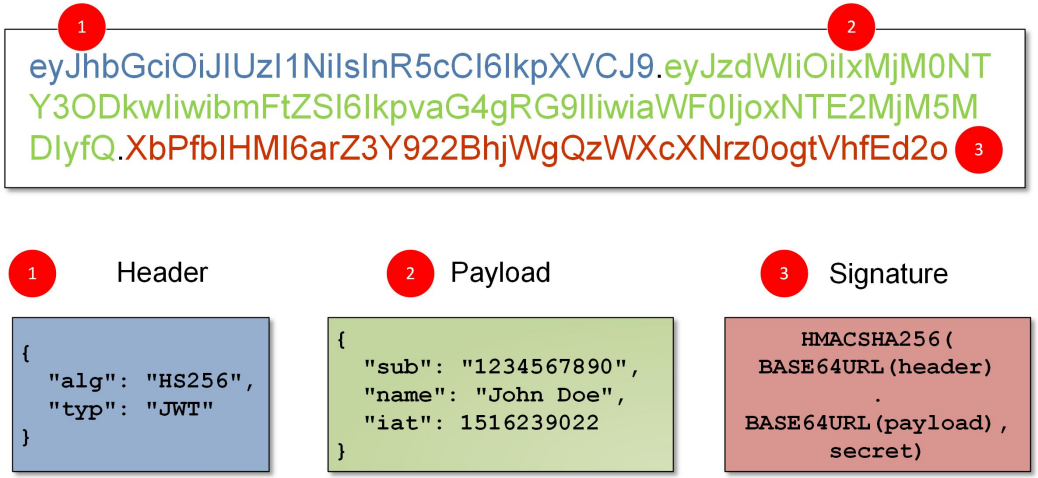
Token Based Authentication Flow





Tokens

- JWT (JSON Web Token)
  - jsonwebtoken library
- JWT guide for Express.js
- Refresh Tokens

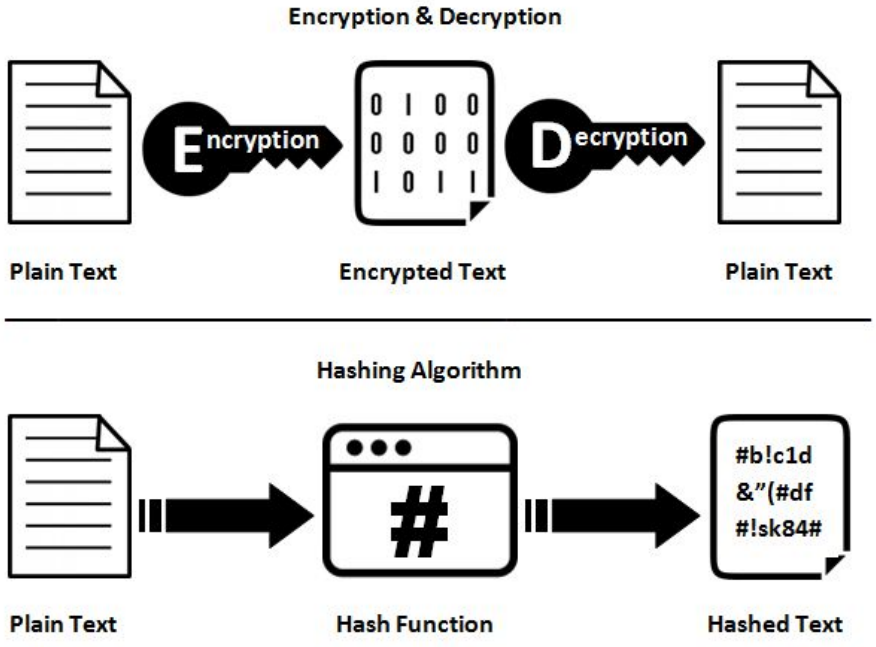


Hashing & Encryption

Hashing & Encryption

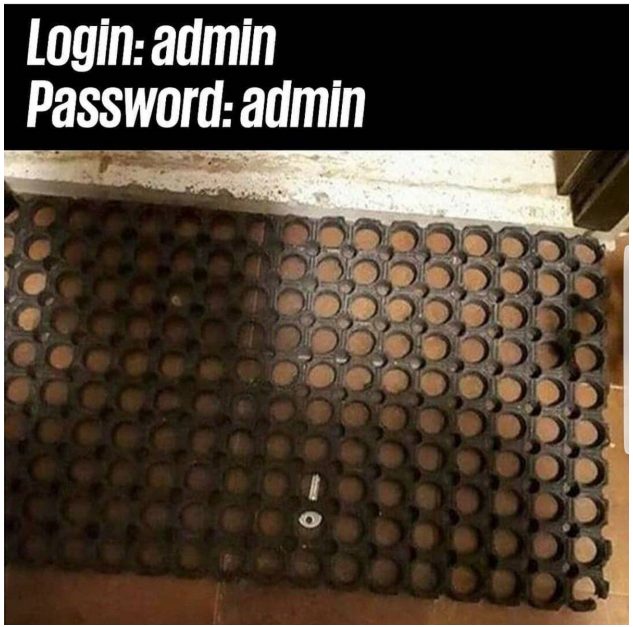
- Use **Hashing** for:
  - Passwords

- **Encryption** is used for:
  - Data Transfer
  - Data at rest
  - Any storage and secure communication.
  - HTTPs



Password storing recipe

- Enforce strong password rules for users
- Never ever store passwords in plain text!
- Use Bcrypt unless you have a good reason not to.
- Use a strong hashing algorithm (never MD5!)
    - Learn how attackers crack password hashes
- Set a reasonable work factor for you system.
- Use a salt
- Make sure that the database password is strong
- Make sure that the production database is encrypted.



Vulnerabilities

Vulnerability scanners

- Known vulnerabilities of dependencies:
  - npm audit
  - Snyk
  - OWASP Dependency Check

- Static security code scanners
  - nodejsscan
  - SonarQube

Secrets Management

Types of secrets

- Database connection strings
- User credentials
- Cryptographic keys
- API keys
- Access tokens
- Cloud service access credentials

Never hardcode these in the codebase!

Never commit them to version control!

Web Server Security

Node.js Security Checklist

- Use appropriate security headers
  - Use a helmet!
  - OWASP Secure headers project
- Perform input validation
- Take precautions against brute-forcing
  - Express-bouncer, express-brute, rate-limiter, captcha
- Only return what is necessary in routes
- Keep your packages up-to-date
- Bookmark this page (Node.js Security Cheat Sheet)

OWASP Top 10 Security Risks

1. **Injection**
2. **Broken Authentication**
3. **Sensitive Data Exposure**
4. **XML External Entities (XXE)**
5. **Broken Access Control**
6. **Security Misconfiguration**
7. **Cross-Site Scripting XSS**
8. **Insecure Deserialization**
9. **Using Components with Known Vulnerabilities**
10. **Insufficient Logging & Monitoring**

OWASP NodeGoat Project

Environment to learn how OWASP Top 10 security risks apply to web applications developed using Node.js and how to effectively address them