Authentication

Nancy Snoke

September 21 2015

Schedule

Date	Talk
September 21, 2015	Authentication / CAS 1
October 5, 2015	Authentication / CAS 2
October 19, 2015	OWASP top 10 – part 1
November 2, 2015	OWASP top 10 – part 2
November 16, 2015	Encryption 1
November 23, 2015	Encryption 2 – bring laptops
December 7, 2015	Spring Security

Warmup Joke


OK, GIVE ME YOUR USERNAME & PASSWORD ONE MORE TIME.
CARTOONSTOCK .com
Search ID: jsh121130
STAHLER.
11/30
©Jeff Stahler/Distributed by Universal Uclick for UFS via CartoonStock.com

RFC 2828 Definition

- Authentication
  - The process of verifying an identity claimed by or for a system entity.

- https://www.ietf.org/rfc/rfc2828.txt

RFC 2828 Definition

- An authentication process consists of two steps:
  - Identification step: Presenting an identifier to the security system.
  - Verification step: Presenting or generating authentication information that corroborates the binding between the entity and the identifier.

What Are The Fours Ways to Verify A User’s Identity?

Ways of Verifying a User's Identity

- Something you know
  - password
- Something you possess
  - Certificate, smartcard, physical key
- Something you are
  - Biometrics, fingerprint, iris
- Something you do
  - Handwriting, voiceprint

Password Based Authentication

- The most common method
- The user provides the username and password
- The system verifies that the password for that username is correct
- The username is used to decide the privileges the user has on the system

What are some password attacks / vulnerabilities?

Password Attacks

- Dictionary attack
- Popular password attack
- Password guessing against particular user
- Browser / Workstation hijacking
  - leaving your computer logged into a application or domain
- User Mistakes
  - Writing down your password on a post it
- Password Reuse
  - Same password multiple accounts
- Shoulder surfing
  - Reading someone's password from over their shoulder

Movie Trivia: Which technique was used in the 1995 movie Hackers by Lord Nikon?

Movie Trivia: Which technique was used in the 1983 classic War Games to get into the WOPR computer?

Password Countermeasures

- Account lockout
- Throttling
- Training on selecting good passwords
- Policies to stop selection of passwords on popular password lists
- Strong password enforcement
- Automatic logout
- Single sign on
- Policies against password reuse

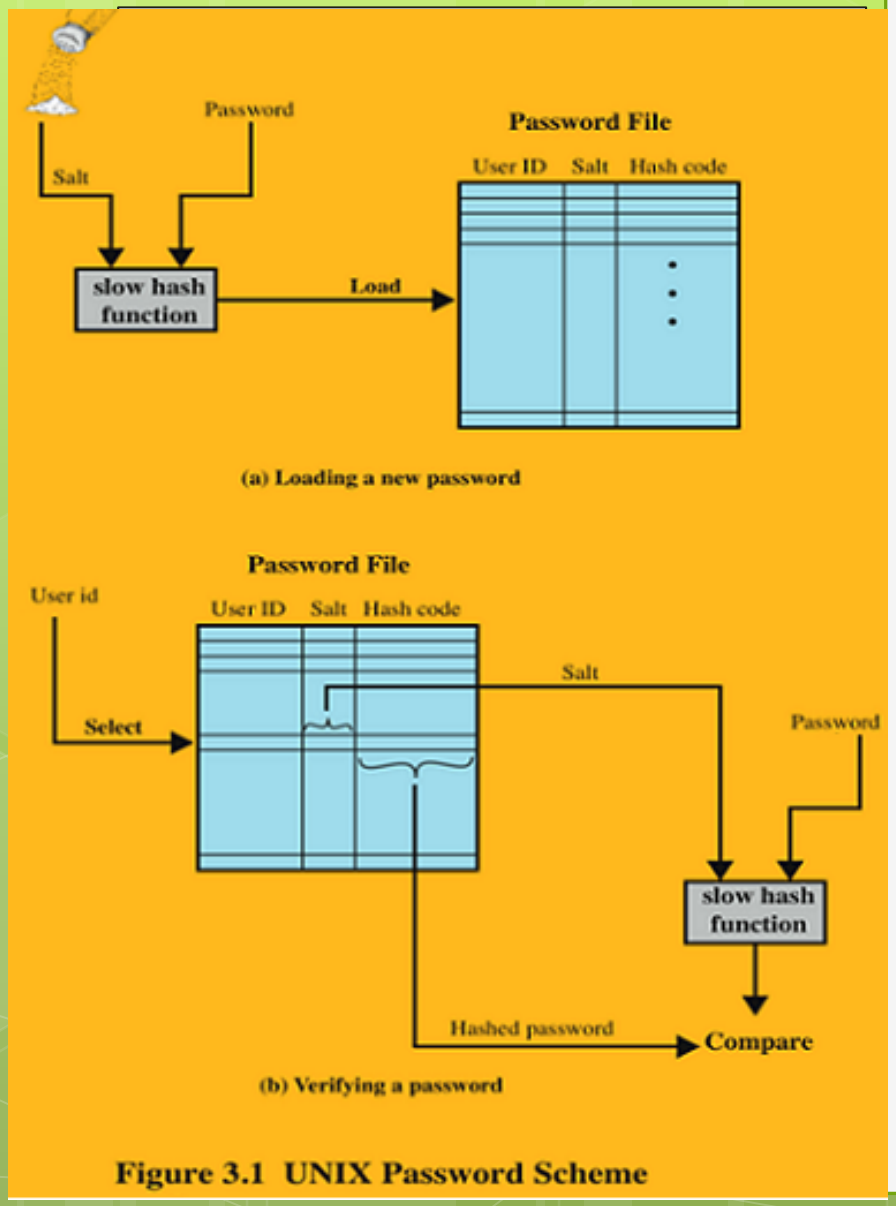

Figure 3.1 UNIX Password Scheme

Hashing Passwords

NIST On Hashing

- National Institute Standards
  - United States Federal Governments
- Hashing Policy as of September 2015
  - Do not use SHA-1 Family
  - SHA-2 is acceptable
  - SHA-3 is acceptable
- http://csrc.nist.gov/groups/ST/hash/policy.html

OWASP On Hashing

- Store a one-way and salted value of passwords.
- Use for Password Storage
  - PBKDF2
  - bcrypt
  - scrypt
- https://www.owasp.org/index.php/Password_Storage_Cheat_Sheet
- https://www.owasp.org/index.php/Cryptographic_Storage_Cheat_Sheet

Contradiction! Help!

- OWASP seems to disagree with NIST
- What should I do?

Closer Look PBKDF2

- DK = PBKDF2(PRF, Password, Salt, c, dkLen)
- *PRF* is a pseudorandom function
  - Such as SHA-1 or SHA-256
- *Password* is the master password
- *Salt*
- *c* is the number of iterations desired
- *dkLen* is the desired length of the derived key
- *DK* is the generated derived key

- https://en.wikipedia.org/wiki/PBKDF2

Why does OWASP recommend these algorithms?

- Iterative
- Can change number of iterations as hardware improves
  - Thus your passwords will continue to be a challenge for hackers
  - Select the number of iterations so that the user impact is not noticeable.

PBKDF2 with SHA-256 would meet both OWASP and NIST standards

Still Not Sure What to Do?

- If you are writing code for the Federal Government follow the NIST standards
- If you have requirements stating to follow all NIST and OWASP recommendations use PBKDF2 with SHA-256
- If this is a personal project in Java use Spring Security's bcrypt
  - PasswordEncoder passwordEncoder = new BCryptPasswordEncoder();
  - String hashedPassword = passwordEncoder. encode(yourpassword);

A Brief Look At CAS

Nancy Snoke

September 21, 2015

What is CAS?

- Central Authentication Service
  - Provides Enterprise Single Sign On
  - Open Source
    - Apache License 2.0
  - Server written in Java
  - Supported clients
    - Java
    - PHP
    - .Net
  - Many Authentication Options
    - Database, LDAP, OTP, X.509, OAuth, and more

CAS History

- Created at Yale
  - By Shawn Bayern
  - Later maintained by Drew Mazurek
- 2004
  - CAS became a project of JASIG
- 2006
  - Yale got Mellon Award for CAS development
- 2008
  - JASIG became responsible for CAS

CAS Protocols

- CAS 1.0
  - single sign on
- CAS 2.0
  - Proxy authentication
- CAS 3.0
  - Easy attribute release

Why Use CAS?

- Free
- Awesome team of developers
- Fantastic community
- Simple to set up simple server
- Any authentication combination you can imagine (practically) can be configured

Prerequisites

- Java SDK 7 or higher
- Apache Maven
- Apache Tomcat
  - Other servlet container could be used
- Basic Spring Framework knowledge
  - Willingness to learn Spring Framework

First CAS Application

- Scenario
  - We have some over powering cute pictures of cats
  - Need to protect people by only allowing authenticated person see these pictures
- Have a Java / JSP based web application
  - Create CAS Server
  - Set up CAS Client in the application

First CAS Server

- Need to allow
  - User: CatAdmirer
  - Password ST.Ciqc!
    - **S**chrodinger **T. C**at **i**s **q**uantum **c**ool **!**
- Unicon provides a simple overlay template
  - https://github.com/UniconLabs/simple-cas4-overlay-template/blob/master/pom.xml

First CAS Server Continued

- Copy pom
  - Set version
  - Remove excludes from pom
- Run mvn clean package
- Copy DeployerContextConfig
  - Modify user
- Run mvn clean package
- Deploy war

Adding CAS Client

- Modify the pom to include CAS Client
  - <dependency> <groupId>org.jasig.cas.client</groupId> <artifactId>cas-client-core</artifactId> <version>${cas.version}</version> <scope>runtime</scope> </dependency>
  - <properties><cas.version>3.4.1</cas.version> </properties>

CAS Authentication Filter

- Determines if a user needs to be authenticated
- <filter>
  - <filter-name>CAS Authentication Filter</filter-name>
  - <filter-class>org.jasig.cas.client.authentication.AuthenticationFilter</filter-class>
  - <init-param>
    - <param-name>casServerLoginUrl</param-name>
    - <param-value>https://127.0.0.1:8080/cas/login</param-value>
  - </init-param>
  - <init-param>
    - <param-name>serverName</param-name>
    - <param-value>http://127.0.0.1:8080</param-value>
  - </init-param>
  - <init-param>
    - <param-name>serviceParameterName</param-name>
    - <param-value>service</param-value>
  - </init-param>
- </filter>

CAS Authentication Filter

- <filter-mapping>
  - <filter-name>CAS Authentication Filter</filter-name>
  - <url-pattern>/protected/*</url-pattern>
- </filter-mapping>

CAS Validation Filter

- Validates the tickets

```
<filter>
    <filter-name>CAS Validation Filter</filter-name>
    <filter-class>org.jasig.cas.client.validation.Cas20ProxyReceivingTicketValidationFilter</filter-class>
    <init-param>
        <param-name>casServerUrlPrefix</param-name>
        <param-value>http://127.0.0.1:8080/cas</param-value>
    </init-param>
    <init-param>
        <param-name>serverName</param-name>
        <param-value>http://127.0.0.1:8080</param-value>
    </init-param>
    <init-param>
        <param-name>redirectAfterValidation</param-name>
        <param-value>true</param-value>
    </init-param>
    <init-param>
        <param-name>useSession</param-name>
        <param-value>true</param-value>
    </init-param>
</filter>
```

CAS Validation Filter

- <filter-mapping>
  - <filter-name>CAS Validation Filter</filter-name>
  - <url-pattern>/*</url-pattern>
- </filter-mapping>

CAS HttpServletRequest Wrapper Filter

- Allows you to use getPrincipal to get the username used for the CAS login

```
<filter>
    <filter-name>CAS HttpServletRequest Wrapper Filter</filter-name>
    <filter-class>org.jasig.cas.client.util.
    HttpServletRequestWrapperFilter</filter-class>
</filter>

<filter-mapping>
    <filter-name>CAS HttpServletRequest Wrapper Filter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

CAS Assertion Thread Local Filter

- Makes the CAS Assertion available.
- Needed if you are returning attributes from CAS.

- <filter>
  - <filter-name>CAS Assertion Thread Local Filter</filter-name>
  - <filter-class>org.jasig.cas.client.util.AssertionThreadLocalFilter</filter-class>
- </filter>

- <filter-mapping>
  - <filter-name>CAS Assertion Thread Local Filter</filter-name>
  - <url-pattern>/*</url-pattern>
- </filter-mapping>

You can try it yourself

- https://github.com/securitymagick/CAS-tutorials/tree/master/tutorial-1

Next Time

- Setting Up SSL to work in development
- Authentication Security Issues
- CAS Security configuration
- Database Authentication