Security Lunch and Learn 2

Nancy Snoke

October 5, 2015

Warmup


Sorry, but your password must contain an uppercase letter, a number, a haiku, a gang sign, a hieroglyph and the blood of a virgin.
someecards
user card

Change to Cat App Demo

Overpowering Cute Cat App DB

- Download MySQL
  - mysql-installer-web-community-5.6.26.0.msi
- Run the installer.
  - Add another account as DBAdmin.
    - catAdmin / catPass.

Database Setup

- From the commandline client:
- CREATE DATABASE insecureCat;
- Use insecureCat;

- CREATE TABLE `catlovers` (
- `cat_name` varchar(45) NOT NULL,
- `email` varchar(45) NOT NULL,
- `uname` varchar(45) NOT NULL,
- `pass` varchar(45) NOT NULL,
- `regdate` date NOT NULL,
- `securityQuestion` varchar(45) NOT NULL,
- `answer` varchar(45) NOT NULL,
- PRIMARY KEY (`email`)
-);

Database Setup Continued

```
CREATE TABLE `catcomments` (
  `id` int(10) unsigned NOT NULL auto_increment,
  `uname` varchar(45) NOT NULL,
  `writtenBy` varchar(45) NOT NULL,
  `comment` varchar(128) NOT NULL,
  PRIMARY KEY  (`id`)
);

CREATE TABLE `catphotos` (
  `uname` varchar(45) NOT NULL,
  `private` varchar(1) NOT NULL,
  `public` varchar(1) NOT NULL,
  PRIMARY KEY  (`uname`)
);
```

CAS Setup For DB

- Modify Pom
- <dependency>
- <groupId>org.jasig.cas</groupId>
- <artifactId>cas-server-support-jdbc</artifactId>
- <version>${cas.version}</version>
- </dependency>
- <dependency>
- <groupId>c3p0</groupId>
- <artifactId>c3p0</artifactId>
- <version>0.9.1.2</version>
- </dependency>
- <dependency>
- <groupId>mysql</groupId>
- <artifactId>mysql-connector-java</artifactId>
- <version>5.1.26</version>
- </dependency>

CAS Setup For DB

- Add to DeployerConfigContext.xml
- <bean id="dataSource"
- class="com.mchange.v2.c3p0.ComboPooledDataSource"
- p:driverClass="${database.driverClass}"
- p:jdbcUrl="${database.url}"
- p:user="${database.user}"
- p:password="${database.password}"
- p:initialPoolSize="${database.pool.minSize}"
- p:minPoolSize="${database.pool.minSize}"
- p:maxPoolSize="${database.pool.maxSize}"
- p:maxIdleTimeExcessConnections="${database.pool.maxIdleTime}"
- p:checkoutTimeout="${database.pool.maxWait}"
- p:acquireIncrement="${database.pool.acquireIncrement}"
- p:acquireRetryAttempts="${database.pool.acquireRetryAttempts}"
- p:acquireRetryDelay="${database.pool.acquireRetryDelay}"
- p:idleConnectionTestPeriod="${database.pool.idleConnectionTestPeriod}"
- p:preferredTestQuery="${database.pool.connectionHealthQuery}" />

CAS Setup For DB

- Add to deployerConfigContext.xml
- <bean id="passwordEncoder"
- class="org.jasig.cas.authentication.handler.PlainTextPasswordEncoder" />

- <bean id="dbAuthHandler"
- class="org.jasig.cas.adaptors.jdbc.
SearchModeSearchDatabaseAuthenticationHandler"
- p:dataSource-ref="dataSource"
- p:passwordEncoder-ref="passwordEncoder"
- p:tableUsers="catlovers"
- p:fieldUser="uname"
- p:fieldPassword="pass" />

CAS Setup For DB

- Modify in deployerConfigContext.xml
- <bean id="authenticationManager" class="org.jasig.cas.authentication.
  PolicyBasedAuthenticationManager">
- <constructor-arg>
- <map>
- <!--
- | IMPORTANT
- | Every handler requires a unique name.
- | If more than one instance of the same handler class is configured, you must explicitly
- | set its name to something other than its default name (typically the simple class name).
- -->
- <entry key-ref="proxyAuthenticationHandler" value-ref="proxyPrincipalResolver" />
- <entry key-ref="dbAuthHandler" value-ref="primaryPrincipalResolver" />
- </map>
- </constructor-arg>

CAS Setup For DB

- Copy cas.properties from target to src
- Add
- # == Basic database connection pool configuration ==
- database.driverClass=com.mysql.jdbc.Driver
- database.url=jdbc:mysql://localhost:3306/insecureCat
- database.user=catAdmin
- database.password=catPass
- database.pool.minSize=6
- database.pool.maxSize=18
-
- # Maximum amount of time to wait in ms for a connection to become
- # available when the pool is exhausted
- database.pool.maxWait=10000
-
- # Amount of time in seconds after which idle connections
- # in excess of minimum size are pruned.
- database.pool.maxIdleTime=120
-
- # Number of connections to obtain on pool exhaustion condition.
- # The maximum pool size is always respected when acquiring
- # new connections.
- database.pool.acquireIncrement=6
-

Cas Set Up For DB

- Add to cas.properties
- # == Connection testing settings ==
-
- # Period in s at which a health query will be issued on idle
- # connections to determine connection liveliness.
- database.pool.idleConnectionTestPeriod=30
-
- # Query executed periodically to test health
- database.pool.connectionHealthQuery=select 1
-
- # == Database recovery settings ==
-
- # Number of times to retry acquiring a _new_ connection
- # when an error is encountered during acquisition.
- database.pool.acquireRetryAttempts=5
-
- # Amount of time in ms to wait between successive aquire retry attempts.
- database.pool.acquireRetryDelay=2000

OWASP Top-10 2013

Dave Wichers
OWASP Top 10 Project Lead
OWASP Board Member
COO/Cofounder, Aspect Security

OWASP
The Open Web Application Security Project

OWASP Top Ten (2013 Edition)

OWASP
The Open Web Application Security Project

A1: Injection	A2: Broken Authentication and Session Management	A3: Cross-Site Scripting (XSS)	A4: Insecure Direct Object References
A5: Security Misconfiguration	A6: Sensitive Data Exposure	A7: Missing Function Level Access Control	A8: Cross Site Request Forgery (CSRF)
	A9: Using Known Vulnerable Components	A10: Unvalidated Redirects and Forwards	

Mapping from 2010 to 2013 Top 10

OWASP
The Open Web Application Security Project

OWASP Top 10 – 2010 (old)	OWASP Top 10 – 2013 (New)
2010-A1 – Injection	2013-A1 – Injection
2010-A2 – Cross Site Scripting (XSS)	2013-A2 – Broken Authentication and Session Management
2010-A3 – Broken Authentication and Session Management	2013-A3 – Cross Site Scripting (XSS)
2010-A4 – Insecure Direct Object References	2013-A4 – Insecure Direct Object References
2010-A5 – Cross Site Request Forgery (CSRF)	2013-A5 – Security Misconfiguration
2010-A6 – Security Misconfiguration	2013-A6 – Sensitive Data Exposure
2010-A7 – Insecure Cryptographic Storage	2013-A7 – Missing Function Level Access Control
2010-A8 – Failure to Restrict URL Access	2013-A8 – Cross-Site Request Forgery (CSRF)
2010-A9 – Insufficient Transport Layer Protection	2013-A9 – Using Known Vulnerable Components (NEW)
2010-A10 – Unvalidated Redirects and Forwards (NEW)	2013-A10 – Unvalidated Redirects and Forwards
3 Primary Changes:	▪ Merged: 2010-A7 and 2010-A9 -> 2013-A6
▪ Added New 2013-A9: Using Known Vulnerable Components	▪ 2010-A8 broadened to 2013-A7

2013-A1 – Injection

OWASP
The Open Web Application Security Project

Injection means…

- Tricking an application into including unintended commands in the data sent to an interpreter

Interpreters…

- Take strings and interpret them as commands
- SQL, OS Shell, LDAP, XPath, Hibernate, etc…

SQL injection is still quite common

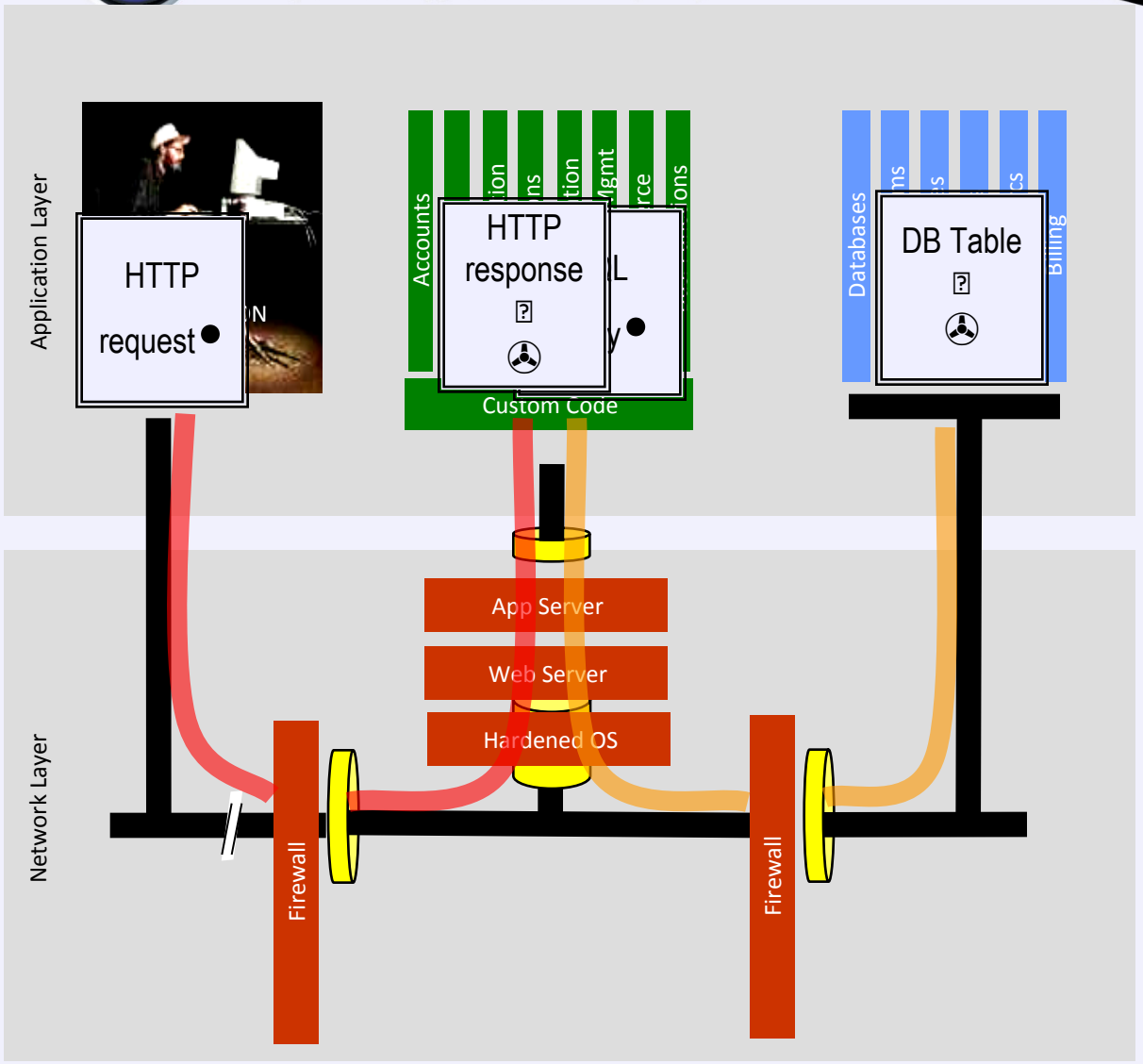
- Many applications still susceptible (really don't know why)
- Even though it's usually very simple to avoid

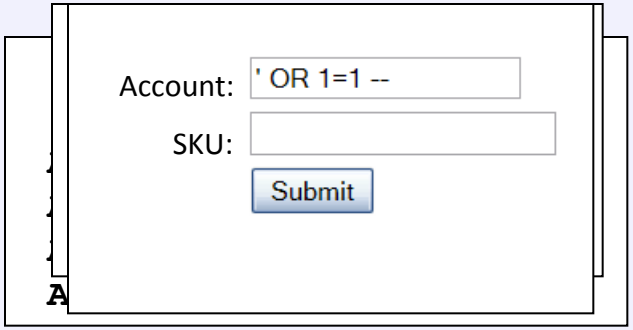
Typical Impact

- Usually severe. Entire database can usually be read or modified
- May also allow full database schema, or account access, or even OS level access

SQL Injection – Illustrated

OWASP
The Open Web Application Security Project





1. Application presents a form to the attacker
2. Attacker sends an attack in the form data
3. Application forwards attack to the database in a SQL query
4. Database runs query containing attack and sends encrypted results back to application
5. Application decrypts data as normal and sends results to the user

Demo SQL Injection

A1 – Avoiding Injection Flaws

OWASP
The Open Web Application Security Project

Recommendations

- Avoid the interpreter entirely, or
- Use an interface that supports bind variables (e.g., prepared statements, or stored procedures),
  - Bind variables allow the interpreter to distinguish between code and data
- Encode all user input before passing it to the interpreter
- Always perform 'white list' input validation on all user supplied input
- Always minimize database privileges to reduce the impact of a flaw

References

- For more details, read the https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

What doesn't work?

- Blacklisting
  - Impossible to blacklist everything
  - Unicode
  - New Attacks
- Hiding the errors
  - Turns it into blind SQL Injection

Fixing Specific SQL Injection

- Statement st = con.createStatement();
- ResultSet rs = st.executeQuery("select securityQuestion from catlovers where uname = '" + user + "'");

- String selectStatement = "select securityQuestion from catlovers where uname = ? ";
- PreparedStatement prepStmt = con.prepareStatement (selectStatement);
- prepStmt.setString(1, user);
- ResultSet rs = prepStmt.executeQuery();

Demo Fixed SQL Injection

2013-A2 – Broken Authentication and Session Management

OWASP
The Open Web Application Security Project

HTTP is a “stateless” protocol

- Means credentials have to go with every request
- Should use SSL for everything requiring authentication

Session management flaws

- SESSION ID used to track state since HTTP doesn’t
  - and it is just as good as credentials to an attacker
- SESSION ID is typically exposed on the network, in browser, in logs, …

Beware the side-doors

- Change my password, remember my password, forgot my password, secret question, logout, email address, etc…

Typical Impact

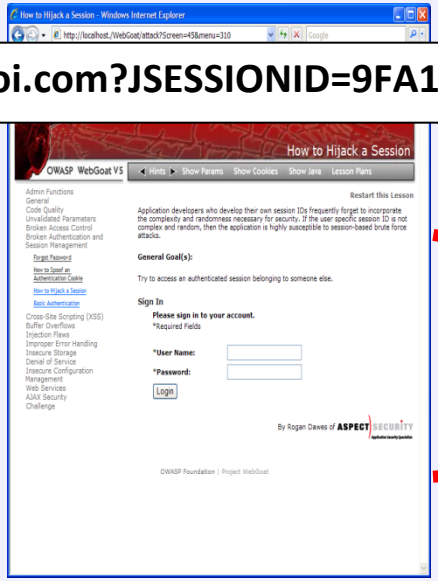
- User accounts compromised or user sessions hijacked

Broken Authentication Illustrated

OWASP
The Open Web Application Security Project

1 User sends credentials

www.boi.com?JSESSIONID=9FA1DB9EA...



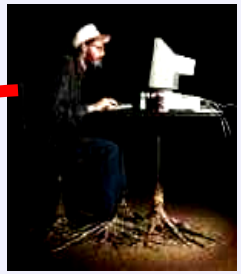
Site uses URL rewriting
(i.e., put session in URL)

2

3 User clicks on a link to http://www.hacker.com in a forum

Hacker checks referrer logs on www.hacker.com
and finds user's JSESSIONID

4



5 Hacker uses JSESSIONID and takes over victim's account

Session Management Demo

A2 – Avoiding Broken Authentication and Session Management

OWASP
The Open Web Application Security Project

Verify your architecture

- Authentication should be simple, centralized, and standardized
- Use the standard session id provided by your container
- Be sure SSL protects both credentials and session id at all times

Verify the implementation

- Forget automated analysis approaches
- Check your SSL certificate
- Examine all the authentication-related functions
- Verify that logoff actually destroys the session
- Use OWASP's WebScarab to test the implementation

Follow the guidance from

- https://www.owasp.org/index.php/Authentication_Cheat_Sheet

What Authentication Issues Do You See in DVOCCA?

Authentication Issues

- SSL
  - Session Cookie set Secure
- Hashed Passwords
- Bad security questions
- No Side channel for password reset
- Enumerate usernames
- Allows same username
- Can browse directly to resetPasswordNow.jsp
  - Do not have to go through security checks

Reset Password with Side Channel

- Gather Identity Data or Security Questions
- Verify Security Questions
- Send a Token Over a Side-Channel
- Allow user to change password in the existing session
- Logging

- https://www.owasp.org/index.php/Forgot_Password_Cheat_Sheet

How Can We Fix Username Enumeration?

How can we fix the allows same username?

How can we fix the direct browsing to resetPasswordNow.jsp?