

Implementation of Secure Hash Algorithm Using JAVA Programming

Revati Raman Dewanagn, Deepali Thombre, Kaushlendra Sharma

¹Asstistant Professor, Computer Science and Engineering , CCET Bhilai, CG,India

² Asstistant Professor, Information Technology, SSEC Bhilai, CG, India

³ Asstistant Professor, Computer Science and Engineering, CCET Bhilai, CG, India

Abstract - In today scenario Security is main issue when we talk about network , network based application and web based application or Web Services like SOAP(Simple object access protocol). Authentication of a user to learn his or her identity. The identity information might be used to make sure a person should have access to the Web services or not. We may also use the identity to track the user's activities to analyse the user responsiveness for many purposes. Here in this paper we have implemented SHA (Secure Hash Algorithm), that is much capable to do this job for secure authentication of user.

Key Words: SOAP, Web Services, Web Application, SHA, Security

1. INTRODUCTION

The Web Application architecture comprises various technologies which enable a client (thin client or web browser) to obtain data from a server, using the http or https protocol. A Web Application provides a web API (application programming interface) which enables two applications to communicate using various method of http over the web, or a network connection. This system was created to act as a middle agent when we talk about the client-server application[1].

A Web Application may be developed in any language and deployed over any platform, but most importantly it may be accessed by any other application regardless of the language used to develop it. Here given below figure is clear snap of web application architecture.

Secure coding is good practice of writing code for applications, systems and web pages in such a way as to ensure the integrity, confidentiality and accessibility of data and information in systems[2]. Web application assessment, ISS/C uses WebInspect, an automated Web application and Web services vulnerability assessment tool that is specifically designed to assess potential security flaws and to provide all the information needed to fix them. As these agents complete the assessment,

findings are reported to a main security engine that analyzes the results. Programmers fluent in secure coding practices can avoid common security flaws in programming languages and follow best practices to help avoid the increasing number of targeted attacks that focus on application vulnerabilities. As an assessment is initiated, WebInspect assigns "assessment agents" that dynamically catalog all areas of a Web application.

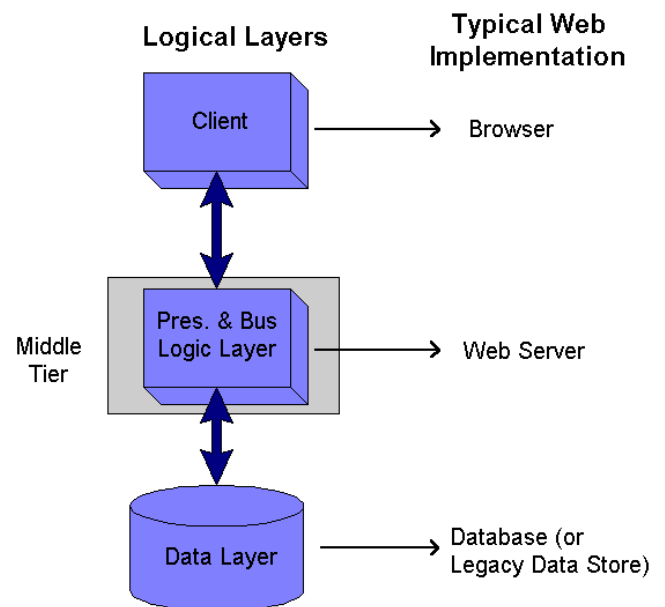


Figure1 : architecture of web application

Manual assessment using WebInspect is also possible for in-depth testing. Reporting is provided in the mail GUI console and as standalone reports in numerous formats. WebInspect then launches audit engines to evaluate the gathered information and apply attack algorithms to locate vulnerabilities and determine their strictness.

1.1 Authentication

This section deals with authentication issues associated with secure web apps, such as basic/digest authentication, form-based authentication, integrated (SSO) authentication, etc.

1.2 Authorization

Topics such as principle of least privilege, client-side authorization tokens, etc. are addressed here. This section addresses authentication issues, ensuring a user has the appropriate privileges to view a resource

1.3 Session Management

This section addresses topics such as authenticated users having a robust and cryptographically secure association with their session, applications enforcing authorization checks and applications avoiding or preventing common web attacks, such as replay, request forging and man-in-the-middle.

1.4 Data Validation

This section deals with applications being robust against all forms of input data, whether obtained from the user, infrastructure, external entities or databases.

1.5 Interpreter Injection

This section addresses application issues so they are secure from well-known parameter manipulation attacks against common interpreters.

1.6 Canonicalization, Locale and Unicode

This section addresses issues that help to ensure the application is robust when subjected to encoded, internationalized and Unicode input.

1.7 Error Handling, Auditing and Logging

This section deals with designing well-written applications that have dual-purpose logs and activity traces for audit and monitoring. This makes it easy to track a transaction without excessive effort or access to the system. They should possess the ability to easily track or identify potential fraud or anomalies end-to-end.

1.8 Distributed Computing

This section deals with synchronization and remote services to web applications, by hardening applications against:

- time of check, time of use race conditions
- distributed synchronization issues
- common multi-programming, multi-threaded and distributed security issues

1.9 Buffer Overflow

This section addresses issues such as:

- Applications do not expose themselves to faulty components
- Applications create as few buffer overflows as possible
- Developers are encouraged to use languages and frameworks that are relatively immune to buffer overflows.

1.10 Administrative Interfaces

This section addresses issues such that:

- Administrator level functions are appropriately segregated from user activity
- Users cannot access or utilize administrator functionality
- Provide necessary audit and traceability of administrative functionality

1.11 Cryptography

This section helps to ensure that cryptography is safely used to protect the confidentiality and integrity of sensitive user data

1.12 Software Quality Assurance (QA)

According to the OWASP 3.0 guide, "The software quality assurance goal is to confirm the confidentiality and integrity of private user data is protected as the data is handled, stored, and transmitted. The QA testing should also confirm the application cannot be hacked, broken, commandeered, overloaded, or blocked by denial of service attacks, within acceptable risk levels. This implies that the acceptable risk levels and threat modeling scenarios are established up front, so the developers and QA engineers know what to expect and what to work towards."

1.13 Deployment

This section deals with the issues surrounding secure deployment of web applications.

1.14 Maintenance

This section addresses issues such as:

- Products are properly maintained post deployment
- Minimize the attack surface area through out the production lifecycle
- Security defects are fixed properly and in a timely fashion

Above all the techniques that are good for building secure web app and secure coding in n-tier architecture.

1.15 Configuration

This section is focused on creating secure web applications which are as well-built and secure out-of-the-box as possible.

2. Critical Security Issues In Web Application

2.1 Broken access control Results from improper enforcement of restrictions on what authenticated users are allowed to do; attackers exploit to access other accounts or use unauthorized functions[3].

2.2 Buffer overflows These components can include CGI, libraries, drivers and Web application server components. Web application components written in languages that do not properly validate input can crash and in some cases, be used to take control of a process.

2.3 Non-validated input Attackers can use information not validated before used by a Web application to reach backend components.

2.4 Cross site scripting A successful attack can disclose the end user's session token or spoof content to fool the user. The Web application is used as a mechanism to transport an attack to the end user's browser.

2.5 Injection flaws If an attacker embeds malicious commands in the parameters, the external system may execute those commands on behalf of the Web application. Web applications pass parameters when they access external system or the local OS.

2.6 Insecure configuration management Having a strong configuration standard is critical. Web servers have many configuration options that effect security and are not secure out of the box.

2.7 Broken authentication and session management Account credentials and session tokens not properly protected, allowing attackers to compromise passwords, keys, session cooker or tokens, and assume the identities of other users.

2.8 Insecure storage Web applications that use cryptographic functions to protect information and credentials have proven difficult to code properly, resulting in weak protection.

2.9 Denial of service Attackers can also lock users out of their accounts or cause an application to fail. As mentioned above, attackers consume Web application resources o the point where other legitimate users can no longer access or use the application.

2.10 Improper error handling Attackers can use these to gain detailed system information, deny service, and cause security mechanisms to fail or crash the server. Refers to error conditions that occur during normal operations that are not handled properly.

3. CRYPTOGRAPHY

Cryptography arose as a means to enable parties to maintain privacy of the information they send to each other, even in the presence of an adversary with access to the communication channel. While providing privacy remains a central goal, the field has expanded to encompass many others, including not just other goals of communication security, such as guaranteeing integrity and authenticity of communications, but many more sophisticated and fascinating goals.

Once largely the domain of the military, cryptography is now in widespread use, and you are likely to have used it even if you don't know it. When you shop on the Internet, for example to buy a book at www.amazon.com, cryptography is used to ensure privacy of your credit card number as it travels from you to the shop's server. Or, in electronic banking, cryptography is used to ensure that your checks cannot be forged.

Cryptography has been used almost since writing was invented. For the larger part of its history, cryptography remained an art, a game of ad hoc designs and attacks. Although the field retains some of this flavor, the last twenty-five years have brought in something new. The art of cryptography has now been supplemented with a legitimate science. In this course we shall focus on that science, which is modern cryptography.

For benefit of community, Humans used the science of cryptography or "secret messages" for thousands of years to transmit and store information needing secrecy. Until recently the military expended most of the effort and money involved[4].

However, starting in 1976 with the introduction in the open literature of public key cryptography by Diffie and Hellman, the non-military and academic pursuit of cryptography has exploded.

The computer revolution has given people the means to use far more complicated cryptographic codes, and the same revolution has made such widespread and complex codes necessary. At the start of a new millennium, even non-technical people understand the importance of techniques to secure information transmission and storage.

Cryptography provides four main types of services related to data that is transmitted or stored:

- Confidentiality: keep the data secret.
- Integrity: keep data secure so that no one can alter it..
- Authentication: be certain where the data came from the original destination and save in proper place.
- Non-repudiation: is the assurance that someone cannot deny something.

3.1 SHA (Secure hash algorithm)

The four SHA algorithms are structured differently and are named SHA-0, SHA-1, SHA-2, and SHA-3. SHA-0 is the original version of the 160-bit hash function published in 1993 under the name SHA: it was not adopted by many applications. Published in 1995, SHA-1 is very similar to SHA-0, but alters the original SHA hash specification to correct weaknesses that were unknown to the public at that time. SHA-2, published in 2001, is significantly different from the SHA-1 hash function.

If we see the history we will observe the different security aspects. In 2005, cryptanalysts found attacks on SHA-1 suggesting that the algorithm might not be secure enough for ongoing use. NIST required many applications in federal agencies to move to SHA-2 after 2010 because of the weakness. Although no successful attacks have yet been reported on SHA-2, it is algorithmically similar to SHA-1. In 2012, following a long-running competition, NIST selected an additional algorithm, Keccak, for standardization under SHA-3. We are focused on the SHA-1 algorithm implementation[5].

3.2 SHA - 1

In cryptography, SHA-1 (Secure Hash Algorithm 1) is a cryptographic hash function designed by the United States National Security Agency and is a U.S. Federal Information Processing Standard published by the United States NIST.

3.3 SHA-1 produces a 160-bit (20-byte) hash value known as a message digest. SHA-1 produces a 160-bit (20-byte) hash value known as a message digest[2]. A SHA-1 hash value is typically rendered as a hexadecimal number, 40 digits long. SHA-1 is a member of the Secure Hash Algorithm family.

Given below figure is a clear snap of SHA-1 working model and details of its hidden route path to achieve the security using the cryptography.

Explanation: One iteration within the SHA-1 compression function: A, B, C, D and E are 32-bit words of the state; F is a nonlinear function that varies; \lll_n denotes a left bit rotation by n places; n varies for each operation; W_t is the expanded message word of round t; K_t is the round constant of round t; \boxplus denotes addition modulo 2^{32} .

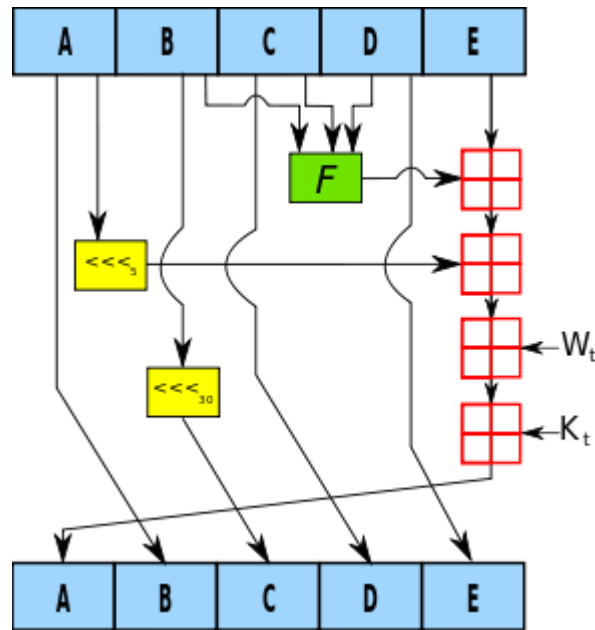


Figure 2- SHA 1 Algorithm

3.4 SHA1 advantages

- There are theoretical attacks against SHA-1 that reduce the difficulty of finding collisions with some other security algorithm. These attacks are still impractical [3], and SHA-1 can be relied on for strong security for hashes and signatures that expire in a few years time duration.
- SHA-2 is similar to SHA-1, but has not been shown to be vulnerable to the same attacks.
- NIST has recommended that the United States Federal Government stop using SHA-1.

3.5 SHA - 1 Implementation

Sun provides SHA1 algorithm in Java under JCE (Java Cryptography Extension) package, which is included in JDK 1.5. Sun's implementation of SHA1 can be accessed through a generic class called MessageDigest[6].

Here are the some main methods of MessageDigest class:

- getInstance("SHA1") - Returns a message digest object represents a specific implementation of SHA1 algorithm.
- getProvider() - Returns the provider name.
- update(bytes) - Updates the input message by appending a byte array at the end.
- digest() - Performs SHA1 algorithm on the current input message and returns the message digest as a byte array.
- reset() - Resets the input message to an empty byte string format.

Here we have implemented SHA - 1 using JAVA Programming Language here is the snap view and prototype of SHA program. The program inherited by java.security and basis class of security features.


```

package project.papaer.sha.sha1;
/**
 * author revati raman and deepali
 * basic classes and interfaces for implementing our
 * methodology is provided by java.security.*; package.
 */
import java.security.*;
class ShaTest7
{
public static void main(String[] a)
{
    try
    {
        MessageDigest mds =
        MessageDigest.getInstance("SHA1");
        System.out.println("Message digest: ");
        System.out.println(" Used Algorithm =
"+mds.getAlgorithm());
        System.out.println(" Provider for the algorithm =
"+mds.getProvider());
        System.out.println(" Convert it toString =
"+mds.toString());
        String input = "";
        mds.update(input.getBytes());
        byte[] output = mds.digest();
        System.out.print("SHA1(\""+input+"\") =");
        System.out.println(" "+bytesToHex(output));
        input = "abcd";
        md.update(input.getBytes());
        output = mds.digest();
        System.out.print("SHA1(\""+input+"\") =");
        System.out.println(" "+bytesToHex(output));
        input = "1234567890";
        mds.update(input.getBytes());
        output = mds.digest();
        System.out.print("SHA1(\""+input+"\") =");
        System.out.println(" "+bytesToHex(output));
    }
    catch (Exception e)
    {
        System.out.println("Exception: "+e);
    }
}

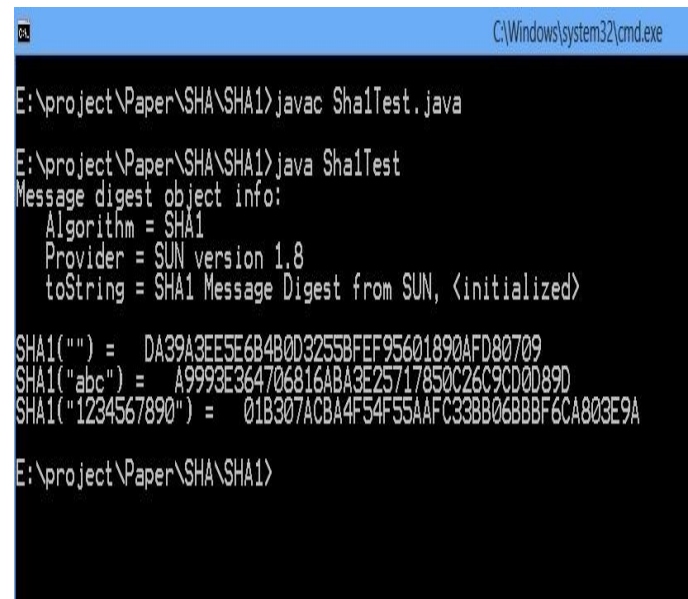
public static String bytesToHex(byte[] b)
{
    char hexDigit[] = {'0', '1', '2', '3', '4', '5', '6', '7',
        '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'};
    StringBuffer buffer = new StringBuffer();
    for (int j=0; j<b.length; j++) {
        buf.append(hexDigit[(b[j] >> 4) & 0x0f]);
        buf.append(hexDigit[b[j] & 0x0f]);
    }
}

//return the elements inside the buffer
return buffer.toString();
}
}

```

4. Result and Discussion

The output matches the testing result available in the following link <http://www.movable-type.co.uk/scripts/sha1.html> . The program is tested by giving the following two different input streams and in both the cases the result of the program and that of in the link was found identical



```

C:\Windows\system32\cmd.exe
E:\project\Paper\SHA\SHA1>javac Sha1Test.java
E:\project\Paper\SHA\SHA1>java Sha1Test
Message digest object info:
Algorithm = SHA1
Provider = SUN version 1.8
toString = SHA1 Message Digest from SUN, <initialized>
SHA1("") = DA39A3EE5E6B4B0D3255BFEF95601890AFD80709
SHA1("abc") = A9993E364706816ABA3E25717850C26C9CD0D89D
SHA1("1234567890") = 01B307ACBA4F54F55AAF33BB06B8BF6CA803E9A
E:\project\Paper\SHA\SHA1>

```

Figure 3. Output and Result of above given implementation

The nature of cryptography and secure hash algorithm will give us the proper way and different approaches to secure our data, application and web pages from unauthorized access. The freedom that we get is to free our self from the outside environment.

We will going to be use this paper to implement in cloud IAAS to secure our cloud data against the unauthorised access in cloud technology. This will secure both of the side server side as well as client side in web application[7].

REFERENCES

- [1] www.asd.gov.au/publications/protect/SHA-1_Deprecated.pdf
- [2] J. Howlader and S. Basu "Sender-Side Public Key Deniable Encryption Scheme/Advances in Recent Technologies in Communication and Computing", *Proceedings of the ARTCom '09 International Conference*, pp.9 -13
- [3] www.h-online.com/.../SHA1-weakness-benefits-password-crackers-1762

BIOGRAPHIES

[4] www.sha1-online.com/sha1-java/

[5] docs.oracle.com/javase/7/docs/api/java/security/MessagesDigest.html

[6] A. O'Neil, C. Peikert and B. Waters Lectute Notes in Computer Science, vol. 6841, pp.525 -542 2011 :Springer - Verlag, Berlin

[7] <http://www.esecurityplanet.com/trends>

[8] Y. Wang, Z. Jin and X. Zhao "Practical defense against wpa and wpa-psk attack for wlan", *Wireless Communications Networking and Mobile Computing (WiCOM), 2010 6th International Conference on*, pp.1 -4

[9] A. El-Din, R. Ramadan and M. Fayek "Vegk: Virtual ecc group key for wireless sensor networks", *Computing, Networking and Communications (ICNC), 2013 International Conference on*, pp.364 -368

[10] Amar Jaffar and Christopher J. Martinez Detail Power Analysis of the SHA-3 Hashing Algorithm Candidates on Xilinx Spartan-3E, *International Journal of Computer and Electrical Engineering*, vol. 5, no. 4, pp.410 -413 2013

[11].Danny Harnik, Benny Pinkas and Alexandra Shulman *Side channels in cloud services, the case of deduplication in cloud storage*, pp.1 -7 2010

[12] Dutch T. Meyer and William J. Bolosky *A Study of Practical Deduplication*, *USENIX Association Berkeley, CA, USA* 2011, pp.1 -1 2012

[13] Thulasimani Lakshmanan and Madheswaran Muthusamy *A Novel Secure Hash Algorithm for Public Key Digital Signature Schemes*, *The International Journal of Information Technology*, vol. 9, no. 3, pp.262 -267 2012

[14] S. Krishnaswamy, W. Hardaker and R. Mundy "DNSSEC in Practice: Using DNSSEC-Tools to Deploy DNSSEC", Conference For Homeland Security, 2009. CATCH '09. Cybersecurity Applications & Technology, pp.3 -15



I Revati Raman Dewangan, Asst. Prof CSE, CCET, Bhilai(CG), total 11 years of experience. 9 years of teaching in Engineering College and 2 years of Software Company. BE in IT 2004 and MTech in CSE 2011. My research area is Data mining, E-Security and Big Data application.



I Deepali Thombre, Asst. Prof. in IT, SSEC bhilai (CG). 5+ years experience of teaching in Engineering College, BE in IT 2009 and MTech in SW engg. 2012. My Research area is e-security and Data Mining.



I Kaushlendra Sharma, Asst. Prof CSE, CCET, Bhilai(CG), total 8+ years of teaching experience in Engineering College. BE in IT 2006 and MTech in CT 2011. My Research area is network and security.