

Ambiguity In Ajax Lockdown Framework

[Unveiling some contradictory facts]

By:Aditya K Sood

Metaeye Security

<http://zeroknock.metaeye.org/mlabs>

Abstract:

Recently I am reading a news at cgisecurity about the latest Ajax lockdown framework proposed by someone. The framework is based on the concept of fusing ajax applications with direct web remoting. The stress is laid on the client server communication and the main point of talk is encrypting the client data and decrypting on the server side. The algorithm blowfish is used and the security is defined on that part. The main point that kept in mind is the traffic intruding, the support for protocol like HTTP/HTTPS is also there with TLS. The security mechanism is bit elaborated on the basis of user data privacy. The concept that is undertaken is the encryption towards the user defined data in the web form mainly where user data privacy is concerned. The data which is incorporated in the web application is encrypted and is stored on the server with the back end defined as the database. I have found some contradictions against this definitive framework which i am going to list in front of the community.

Contradiction A:-

The key used for the encryption or decryption is stored in the client browser itself. It is not being transported to the server. We know that the blowfish algorithm is used where keys are not changed oftenly. It means the specific user holds the credential of some kind which remain present in the browser. There is no specific procedure defined for storing key of the user. The specification provided by the framework is it remain in the client browser. The attacker can easily steal because the key protection mechanism is not defined in a well suited manner. The key called to be as secure key but no secure mechanism has been undertaken. So it becomes very easy for the attacker not to hit the traffic but the client machine where key is stored in the browser. Lot of browser hijack attacks can be formed or Third party attacks can be launched. The information can be easily extracted out of the browser. The normal HTML forms are used for transaction.

[No Secure Method For Handling Client Key]

Contradiction B:-

There is no talk about the data integrity checks i.e. no stress has been provided on how to check the data that is set for input by the user. This is a point of concern because if no input validation check is performed, the rogue data will get automatically encrypted and sent to server for storage, which itself don't know what it is going to be stored on it. This is an issue of great importance too with respect to data integrity. There is no direct talk about sanity checks for the blowfish algorithm as:

```
int Blowfish_Test(BLOWFISH_CTX *ctx)
{
    uint32_t L = 1, R = 2;

    Blowfish_Init(ctx, (unsigned char*)"TESTKEY", 7);
    Blowfish_Encrypt(ctx, &L, &R);
    if (L != 0xDF33FD2L || R != 0x30A71BB4L) return (-1);

    Blowfish_Decrypt(ctx, &L, &R);
    if (L != 1 || R != 2) return (-1); return (0);
}

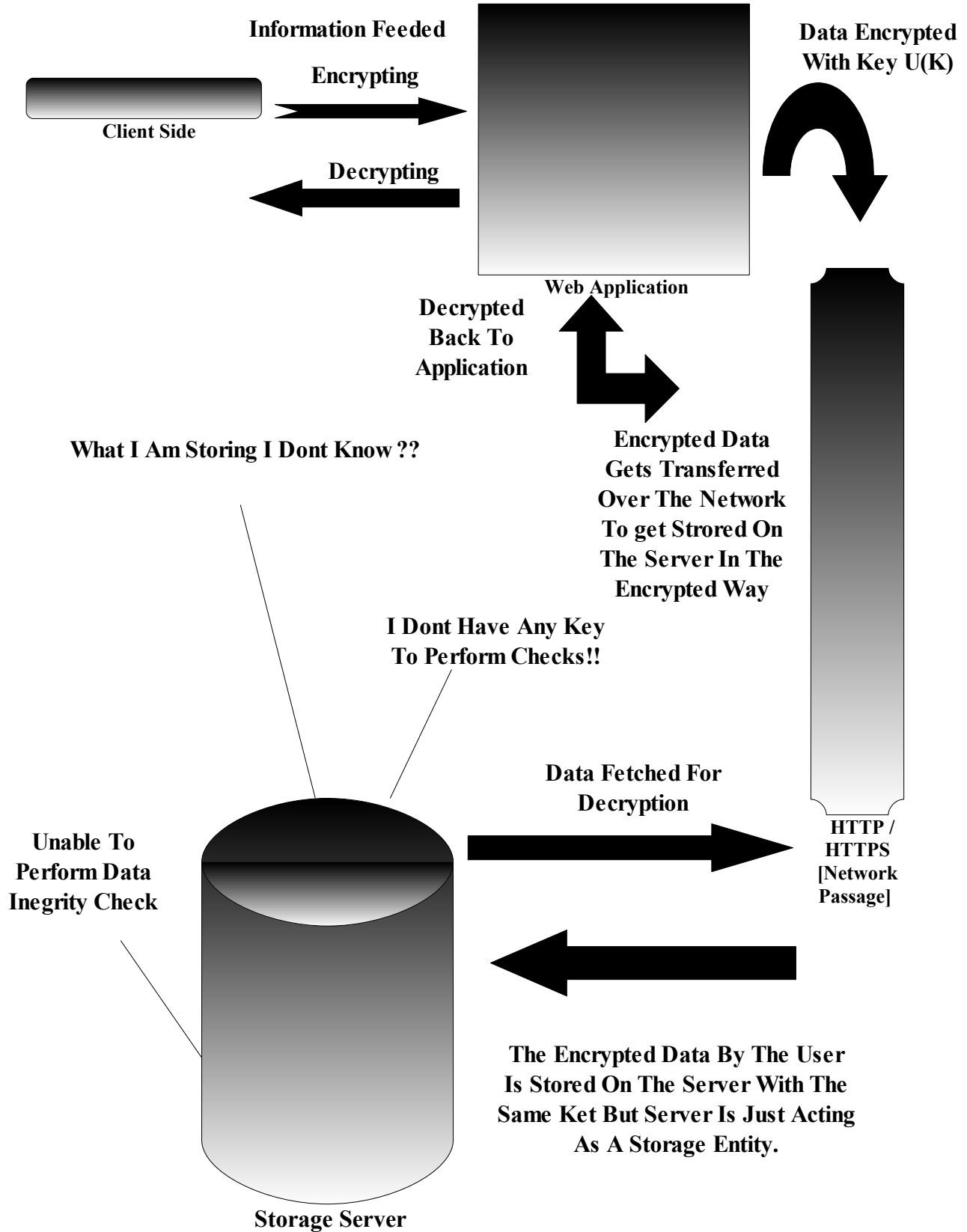
#if 1
    /* sanity test, encrypts a known message */
    n = Blowfish_Test(&ctx);
    printf("Blowfish_Test returned: %d.%s\n", n, n ? " Abort." : "");
    if (n) return n;
#endif
```

There is no talk except the encryption. What if rogue data gets stored in the server. This is not considered to be as the exact way of validating inputs.

[No Exact Way Input Validation Checks Are Undertaken]

[Ambiguity In Ajax LockDown Framework]

[The Ambiguity Layout]



[Ambiguity In Ajax LockDown Framework]

Contradiction C:-

The framework concept is based on the esecure key. So underlined is the key point of the esecure key that I have undertaken from the esecurekey website: It says:

The basic Concept of eSecureKey application is very simple.

- 0xa] The application relies on client side JavaScript engine present in any modern web browser to encrypt data using a Secure Key.
- 0xb] This Secure Key is a user provided key on which the data is encrypted and then saved in the server. This Secure Key is never transported or saved in the server. **So a forgotten Secure Key means complete loss of data which were encrypted with that Secure Key.**
- 0xc] To view the saved data, encrypted data is got back from the server database, the user then need to enter the correct Secure Key so that the saved data can be decrypted and shown.

Once encrypted with a Secure Key the data is safe to be saved anywhere in the web as long as the user remembers the Secure Key. No one else can get access to the data without the Secure Key.

Ofcourse if we dont forget than Human is the weakest element in security. This has made the things complex rather easy because if by mistake user forget the key than what ever he do he is not able to get the data back. This is something out of the way for normal user to be so specific. This is rather increasing complexity.

Contradiction D:-

The server dont know what is present on the it. At first the server sometimes uses its own key to provide security to data.

The framework says:-

"

To configure the database table using Hibernate, I added UserData.java, a wrapper for the underlying saved data in the database having a properties id (unique to each UserData), the data (the encrypted data), and a userid (the user identification ID associated with each UserData); UserData.hbm.xml is defined to map the UserData to a valid table in the database using Hibernate. UserDataFactory.java is used to provide basic functionality like add, edit, delete, and search on the UserData.

A skeletal view of the class is:

```
public class UserDataFactory {  
    Session hibernateSession = null;  
    public UserDataFactory() throws Exception  
    protected void init() throws Exception  
    public UserData getUserData(String dataId)
```

[Ambiguity In Ajax LockDown Framework]

```

public java.util.List getLoggedInUserDataSet(String userid)
public void addUserData(String emailId,String userdata)
public void updateUserData(String id,String emailId, String userData)
public int deleteUserData(String id)
protected void finalize() throws Throwable }
"

```

The hibernate framework has been used for the database and all transaction is carried through the AJAX and Direct Web Remoting. Again the point of talk is that mapping is done through the static xml file that gets processed when the requests is being undertaken by the container. The article written by me ie "Rogue XML Specifications " cover the security and attacks in all and all. The point of concern is if user provide data on the form with a large number of input entities. Lets check the mapping:

```

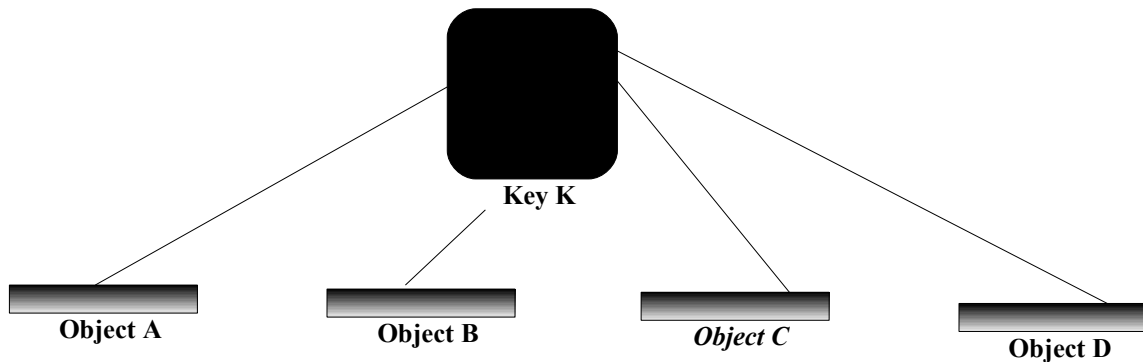
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping package="eg">
</hibernate-mapping>

```

The Problem Area For Hibernate Framework is underlined :

- 0xa] A unique constraint violation could occur if two objects are both being updated, one is "releasing" a value and the other is "obtaining" the same value.
- 0xb] A lot of problem in mapping the <key-many-to-one> tag which is used to map the multiple objects to single object for database transactions.



The objects here can be tables , individual rows , columns etc.

- 0xc] There is no satisfactory implementation of Primary key on tables. The only way to work over columns is to map the <composite/id> in the XML file which rises complexity.
- 0xd] The Hibernation is prone to JDBC connection leakage which attacker can manipulate easily and data can be inadvertently destroyed or wrongly mapped.

0xe] The hackers can easily temper the Hibernate with MYSOL work layout by manipulating the useServerPrepStmts=true which disrupt the functioning of hibernate with MySQL.

These are some of the problem related to the Hibernate framework. The storing of encrypted data without server intervention is much of complex issue. This enhances the complex behavior of working functionality of this framework.

Conclusion:-

Fusing in wrong sense of technology results in complexity and further inadvertent issues which become hard to define. The security when comes to user instance than the security vector from the user perspective should be minimum because this rises the complexity and for normal users it is hard to undertake the complexity and further indepth, is the way to handle this complexity.