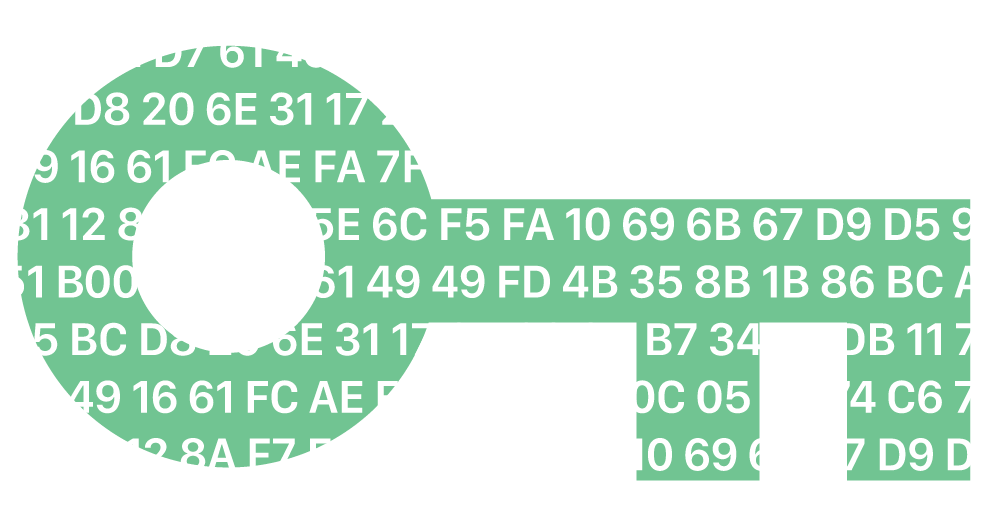
Analyzing Cryptographic Failures



Dimitar Rebranin

College of Engineering and Technology

Utah Valley University

10/10/2024

IT-6780-701 – Secure Coding

Prof. Kodey Crandall

Table of Contents

[1 Key Points of A02:2021 4](#_Toc179447364)

[2 Scenario Analysis: Scenario #2 5](#_Toc179447365)

[3 Practical Application 5](#_Toc179447366)

# 1 Key Points of [A02:2021](https://owasp.org/Top10/A02_2021-Cryptographic_Failures/)

The main purpose of this document is to present the importance of using secure protocols, the right cryptographic algorithms, and encrypting data. It details how dangerous it could be if cryptographic failures happen. The first point made is to identify the data – which data is sensitive according to privacy laws and regulations like GDPR, PCI DSS and such. Once we have identified the sensitive information, we need to decide what data can be stored and if there is no need for this data, just discard it. An attacker cannot tamper with data that is not there.

Next, it is extremely important to understand that the data we decided to keep at rest needs to be encrypted. Encrypting your data will prevent an attack of being able to use or read it. Encrypted data is pretty much unusable unless you know how to decrypt it. This adds an extra layer of security that even stolen data is still safe. Also, encrypt data in transmission. Use secure protocols like HTTPS, TLS, etc. Stay away from the old protocols that are insecure and very outdated and vulnerable – SMTP, HTTP, FTP. These protocols transfer data in plain text and a simple eavesdrop on the network traffic will expose passwords, credit card numbers, etc.

Another point that we can take out of this document is that using the latest and secure cryptographic algorithms is a must. The old schemes like MD5, SHA1 and PKCSv1.5 are now deprecated and very vulnerable. Latest algorithms use much larger key sizes which lead to longer encryption/decryption times, requirements for more processing power thus making them close to impossible to break with rainbow tables, dictionary words and so on. They are more secure than the deprecated ones.

Three common weaknesses that are frequently associated with cryptographic failures:

* Use of hard-coded password
* Broken or Risky Crypto Algorithm
* Insufficient Entropy

# 2 Scenario Analysis: Scenario #2

Couple of days ago, in our other class INFO-6420, we did an exercise where we used the webapps.net vulnerable site and did session hijacking. Because the site does not use HTTPS for its communication between client and server, an attack can eavesdrop on the network traffic and steal the information they needed. In our case, we used Burpsuite and looked at the http headers where we were able to catch the session cookie in plain text. Once we had this cookie id, we used it to log in to the admin user and modify some orders and do some harm.

The solution to scenario #2 on OWASP-A02 could be few things. The first that comes to my mind is to use **HttpOnly** and **Secure** flags which will ensure that the session cookies are only transmitted over encrypted connection. The website in general should be using HTTPS protocol when creating or signing in to an account. That will protect the username and password, and other sensitive information from being transmitted in plain text. Some pages to look at on the OWASP website are [HTTP Security Response Headers](https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Headers_Cheat_Sheet.html) and [Cross Site Scripting Prevention](https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html).

# 3 Practical Application

Even though I am not a web developer, I do some scripting at work, like bash and terraform. The examples provided in this document lead me to rethink the way I write code. I was just looking through some of my old scripts yesterday, and I found a script where I update CMDB fields for 100s of nodes at ones. Well, in the script, every time you PUT into a CMDB entry, you must authenticate the systems. To make life easier, I hardcoded my password there. Even though this password doesn’t exist anymore, if someone got a hold of my script, they will be able to see an example of what I use for passwords. This is like the first example of what NOT to do.

Couple of practices to implement to prevent cryptographic failures:

* use environment variables
* Encrypt all sensitive data at rest
* Encrypt all communications transportation