UNDERSTANDING THE IMPORTANCE OF SECURITY BY DESIGN

CHALLENGE DESCRIPTION

24.09.2019

European Cyber Security Challenge 2019

Bucharest, Romania

1. Initial Write-Up

Description: Security by design is an important measure to counter malicious attacks on vulnerable assets. Coding and implementing should be carried on with a security minded philosophy to prevent breaches based on weak or faulty algorhytms.

This is a simple static code analyzing challenge, where participants have to bypass a serial number checking algorithm by generating valid serial numbers.

The goal is to generate valid serial numbers that can pass the checking function.

2. Challenge specifications

- Category: PHP/code analyzing
- Difficulty: Easy
- Expected time to solve: less than 2 hours

3. Technical specifications

Description: The challenge is implemented through a PDF file, containing the data to be analyzed

Challenge Technical Specification, data to set up and access to the environment.

The challenge is static: distribute the 'Challenge' PDF to participants to analyze the code

- Required language skills: PHP
- Software used
    - PHP

4. Questions and answers

Description:

1. CTF Specific questions:

- What is the used hashing algorithm in the code? **(MD5)**
- How big is the random space in the code? (**64K**)
- Why is this code vulnerable? **(Due to limited randomness it is possible to write a code to simply bruteforce the serial)**
- How long are the accepted serials? **(16 characters)**

2. Non-Flag specific:

- What is entropy? **(In computing, entropy is the randomness collected by an operating system or application for use in cryptography or other uses that require random data)**
- Why is high entropy important in a pseudorandom algorithm? **(To avoid simple bruteforcing and prevent guessing of the valid serials)**
- What are the risks of a client-side serial number validating method? **(A malicious attacker can simply reverse the hashing algorithm, and gets the possibility to validate, and create valid pwd's without the server side being attacked.)**
- What are the security advantages of server-side serial number validation? **(It makes the attacker to try brute-force online, which is slower, and servers can be set to deny the service from them after a few failed attempt)**
- Give at least two ideas how to harden a serial number validating algorithm? **(Server side validation is a better choice, and the obfuscating of the code is also a recommendation to avoid reverse engineering. Third possible answer is to use a higher entropy by setting the random space to 2048k or higher)**

5. Attack Scenario

Description:

This is a simple static code analyzing challenge, where participants have to bypass a serial number checking algorithm by generating valid serial numbers.

6. Installation instructions

Description:

The validating function can be run on a command line interface with PHP either on Linux or on Windows systems.

7. Tools needed

Description:

Tools needed for the solution of the challenge

- Text editor
- PHP

8. Artifacts hashes

File	MD5	SHA256
Challenge.pdf	dc2e2cd1253c0efc9ab2b593d85e0752	fc427204417a15b54eefca82c194ae835b95838868509e3431674fdd5bb4f297
Solution.zip	a91fc4cc6c371d60b4316c79d8e6964b	32c2cda6526fc8ecff7316f51f1bfa79466079c3ec8ebba5edae5213ba2995c8

9. Walkthrough (writeup)

Description:

After a short code-review, it can be recognized that one of the requirements of a valid serial number is to be 16 characters long.

```
    }
    if (($final=="fd56") && (strlen($pwd)==16)) return true;
    return false;
```

The other property of a valid serial number is that variable called $final has to be “fd56”.

```
    if (($final=="fd56") && (strlen($pwd)==16)) return true;
    return false;
```

This property can have no more than 16^4 + 1= 65536 different values. As this is a relatively small entropy, the easiest way to generate a valid serial number is to randomly try 16 characters long strings, and validate them with the provided serial checking algorithm.

```
<?php
ini_set('error_reporting', E_ALL & ~E_NOTICE & ~E_STRICT & ~E_DEPRECATED);

function VSSG($pwd)
{
        $hash= hash('MD5',$pwd);
        for ($x = 0; $x < 8; $x++)
        {
                $xored[$x]=substr($hash,$x*4,4);
        }
        foreach ($xored as $v)
        {
                $final=dechex(hexdec($v) ^ hexdec($final));
        }
        if (($final=="fd56") && (strlen($pwd)==16)) return true;
        return false;
}

function GenerateString()
{
        return substr(str_shuffle("ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"), 0, 16);
}

while(true) {
        $valid = GenerateString();
        if (VSSG($valid))
        {
                echo "Valid Serial is : ".$valid."\n";
        }
}
?>
```

The final serial generating script can be run from command line with php:

```
@~/OneDrive Documents/ENISA/025_PHP_serial/Solution$ php find_serial.php
Valid Serial is : FOZMV7W4IT6D51HL
Valid Serial is : LQ5YNFAMCEDB786J
Valid Serial is : DW8FX53N26EBLPCA
Valid Serial is : VCIZA5H91XE4UNQM
Valid Serial is : V7TWHCPUXZR0JFIQ
Valid Serial is : UCRLXE81FAZSD0GN
Valid Serial is : SYKP6DR4825GEI0F
Valid Serial is : Y7UCR9P23HKNMBDA
```