彻底搞懂base64加解密原理和隐写技术

原创

Slient-猿 于 2020-07-17 15:54:23 发布 708 收藏 2
分类专栏： CTF 文章标签： base64 base64隐写 base64编码原理 base64解码原理 base64隐写原理

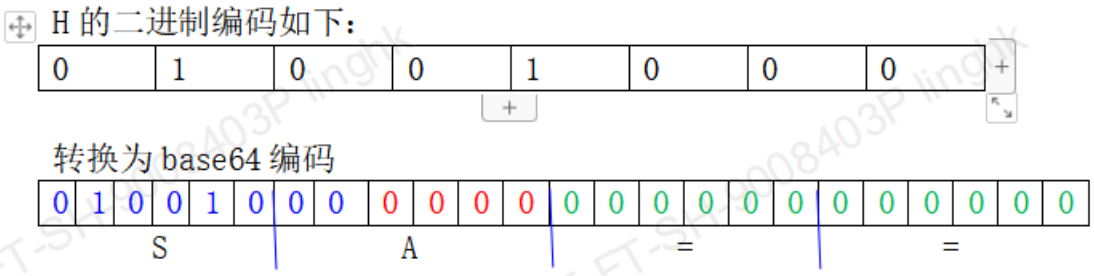
本文链接：https://blog.csdn.net/m0_37948170/article/details/107387926

base64编码和解码是一个常用的方式，可以避免明文传输或者存储，也可以结合加解密技术进行使用。

- base64 编码的定义：base64编码表：

索引	对应字符	索引	对应字符	索引	对应字符	索引	对应字符
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v		
14	O	31	f	48	w		
15	P	32	g	49	x		
16	Q	33	h	50	y		

- base64 采用6位二进制进行编码，不足部分补足0，字符是8位二进制表示，所以最小公约数位24，3位字符经过base64编码后变成了4位。
- 看一个编码示例：



- 从示例可以看到：H是八位，其前六位010 010对应18,18对应base64编码表为S，H的后两位00加上补充的4个0组成6位，正好是000 000 对应十进制为0，对照base64编码表为A，后续补充的0用=号补足，=代表补充二进制位数*2，故最多补充2*2位，两个==代表补足了4个0.一个=号代表补足了两个0.

- base64编码步骤：字符串转换为二进制H(01001000)==》从前到后每六位二进制转换为十进制，去base64编码表查找对应的字符或符号010 010（S）000 000（A）==》根据最后一次补充了的0，对应添加0个或一个或两个=号 0000（==）==》完成编码（SA==）。
- base64解码：从前往后对照base64编码，依次转换为二进制SA（010010 000 000）==》遇见=则结束==》然后从前到后8位二进制转换为字符H（0100 1000），不足8位则丢弃。==》完成转换SA==（H）。
- java 实现的 base64编码解码：

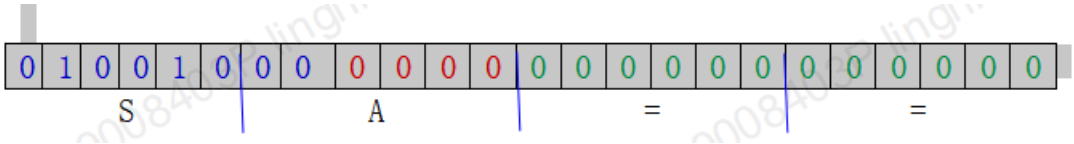
```
public class Test {

    private final static char pem_array[] = {
//           0   1   2   3   4   5   6   7
            'A','B','C','D','E','F','G','H', // 0
            'I','J','K','L','M','N','O','P', // 1
            'Q','R','S','T','U','V','W','X', // 2
            'Y','Z','a','b','c','d','e','f', // 3
            'g','h','i','j','k','l','m','n', // 4
            'o','p','q','r','s','t','u','v', // 5
            'w','x','y','z','0','1','2','3', // 6
            '4','5','6','7','8','9','+','/'  // 7
    };

    private final static byte pem_convert_array[] = new byte[256];

    static {
        for (int i = 0; i < 255; i++) {
            pem_convert_array[i] = -1;
        }
        for (int i = 0; i < pem_array.length; i++) {
            pem_convert_array[pem_array[i]] = (byte) i;
        }
    }

  /**
     * @param str  待编码字符串
     * @return base64 编码字符串
     */
    static String getBase64FromStr(String str){
        String strToTwo = strToTwo(str);
        StringBuilder base64Str=new StringBuilder(str.length());
        for (int i = 0,len=strToTwo.length(); i <strToTwo.length() ; i=i+6) {
            if(i+6>len){
                String sub = strToTwo.substring(i, len);
                int count=0;
                while (sub.length()<6){
                    sub=sub+"0";
                    count++;
                }
                base64Str.append(pem_array[Integer.parseInt(sub,2)]);
                if(count>2)return base64Str.append("==").toString();
                return base64Str.append("=").toString();
            }
            String sub = strToTwo.substring(i, i + 6);
            base64Str.append(pem_array[Integer.parseInt(sub,2)]);
        }

        return base64Str.toString();
    }


 /**
```

```
    * @param str  base64编码字符串
    * @return base64解码字符串
    */
   static String getStrFromBase64(String str){
       int len = str.length();
       char[] chars = str.toCharArray();
       StringBuilder resStr=new StringBuilder(len);
       for (int i = 0; i <len ; i++) {
           if(chars[i]!='='){
               int res=Test.pem_convert_array[chars[i]];
               String binStr = Integer.toBinaryString(res);
               while (binStr.length()<6)binStr="0"+binStr;
               resStr.append(binStr);
           }
       }

       int length = resStr.length();
       StringBuilder retStr = new StringBuilder(len);
       for (int i = 0; i +8<= length; i=i+8) {
           String subStr = resStr.substring(i, i + 8);
           retStr.append((char)Integer.parseInt(subStr,2));
       }
       return retStr.toString();
   }
static String strToTwo(String str){
       char[] chars = str.toCharArray();
       StringBuilder res=new StringBuilder(chars.length*8);
       for (int i = 0,len=chars.length; i <len ; i++) {
           String binStr = Integer.toBinaryString(chars[i]);
           while(binStr.length()<8)binStr="0"+binStr;
           res =res.append(binStr);
       }
       return res.toString();
   }


}
```

- 根据base64的解码过程，我们可以看到隐写的地方。即补充的四个红色0，但是这里最多隐藏4位二进制，这代表base64编码后一个=号可以隐藏最多2位，0个无法隐藏，四个红色0的部分可以被修改，而不影响base64解码，这就是base64隐写的原理。
- 所以隐藏信息需要很多行，一行base64最多隐藏4位，那么我们隐藏一个who。
- who的ascll 二进制编码：0111 0111 01101000 01101111 总共24位那么需要至少6行。如果都用H来隐藏，那么如下可以看到一个示例：

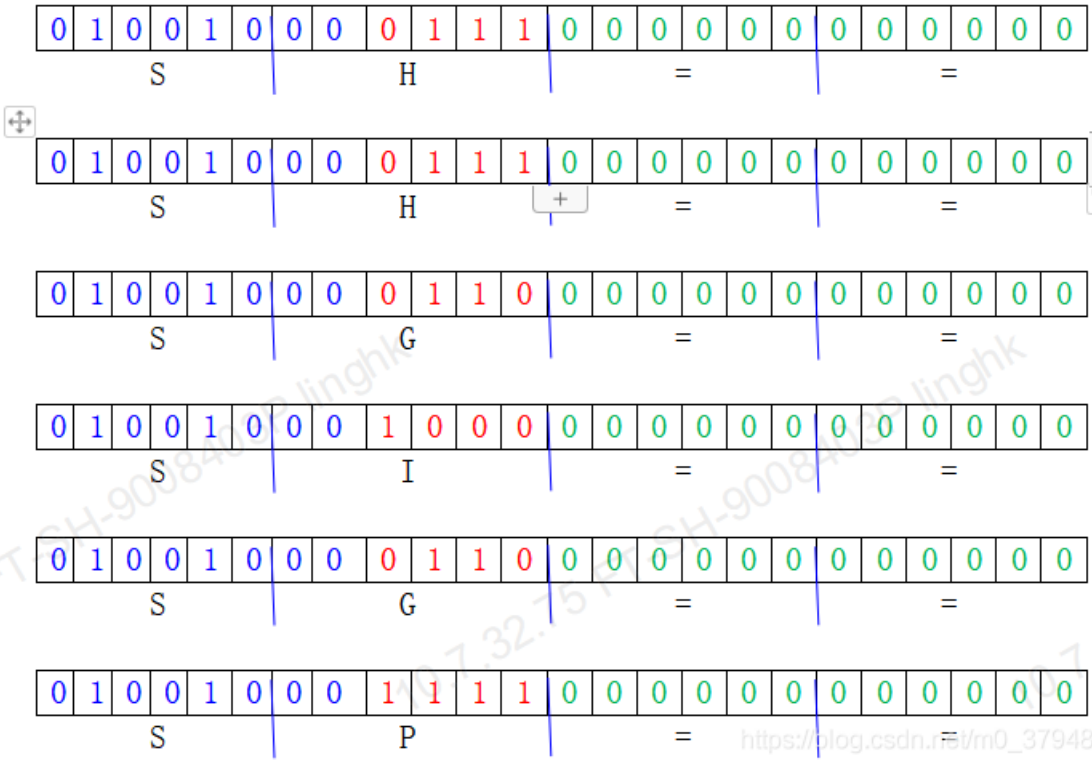
  需要隐藏的who的ASCII码的二进制：0111 0111 01101000 01101111

  H的base64编码：SA==



  需要隐藏的二进制：0111 0111 0110 1000 0110 1111

开始隐藏：



故隐写后的base64编码如下:

SH==

SH==

SG==

SI==

SG==

SP==

给出一个base64隐写解码的java方法：

```
public class Test {

    private final static char pem_array[] = {
//          0   1   2   3   4   5   6   7
            'A','B','C','D','E','F','G','H', // 0
            'I','J','K','L','M','N','O','P', // 1
            'Q','R','S','T','U','V','W','X', // 2
            'Y','Z','a','b','c','d','e','f', // 3
            'g','h','i','j','k','l','m','n', // 4
            'o','p','q','r','s','t','u','v', // 5
            'w','x','y','z','0','1','2','3', // 6
            '4','5','6','7','8','9','+','/'  // 7
    };

    private final static byte pem_convert_array[] = new byte[256];

    static {
        for (int i = 0; i < 255; i++) {
```

```
            pem_convert_array[i] = -1;
        }
        for (int i = 0; i < pem_array.length; i++) {
            pem_convert_array[pem_array[i]] = (byte) i;
        }
    }


    public static void main(String[] args) {

         String steTxt="SH==\n" +
                "SH==\n" +
                "SG==\n" +
                "SI==\n" +
                "SG==\n" +
                "SP==";
        getSteFromBase64(steTxt);
    }


     /**
     *
     * @param stegotextBase64 含隐写的base64加密串
     * @return 隐写的字符串
     */
    static String getSteFromBase64(String stegotextBase64) {
        String[] split = stegotextBase64.split("\n");

        Decoder decoder = Base64.getDecoder();
        StringBuilder base64Str = new StringBuilder(stegotextBase64.length());
        String reSub = "";
        for (int j = 0, len = split.length; j < len; j++) {
            String stegb64 = split[j];
            String rowb64 = getBase64FromStr(getStrFromBase64(stegb64));
            String reStegb64 = stegb64.replace("=", "");
            String reRowb64 = rowb64.replace("=", "");
            int offset = Math.abs(pem_convert_array[reStegb64.toCharArray() [reStegb64.length() - 1]]
                    - pem_convert_array[reRowb64.toCharArray()[reRowb64.length() - 1]]);
            int count = stegb64.length() - reStegb64.length();
            if (count > 0) {
                String binStr = Integer.toBinaryString(offset);
                while (binStr.length() < count * 2) binStr = "0" + binStr;
                reSub = reSub + binStr;
            }
        }
        int length = reSub.length();
        String retStr = "";
        for (int j = 0; j + 8 <=length; j = j + 8) {
            retStr = retStr + (char) Integer.parseInt(reSub.substring(j, j + 8), 2);
        }

        System.out.println(retStr);

        return retStr;
    }

}
```

根据上述过程给出一个可以自己写一个隐写的方法。

总结一下：base64编码原理就是通过使用base64编码表将8位二进制位表示的字符串编码为6位表示的base64编码表的字符，不够6的倍数的通过补充0的方式补足6位，补0的个数用=号表示，一个=代表补两个二进制位0，故经过base64编码后3个8位表示的ASCII 字符就变成了4位base64编码。

解码原理：base64字符串按照base64编码表转换为6位二进制位，然后拼接成字符串，将二进制字符串从前到后按8位进行转换为对应的ASCII码字符，最后不足8位的丢弃即可。

隐写原理：通过base64编码解码原理可以得出：补0的部分不影响解码，故可以利用这些补的二进制位0进行隐写，即将需要隐写的字符串转换为二进制位，去替换补0的部分，达成隐写的目的。一行base64编码最多两个=号，那么只能隐写4位二进制位，隐写字符串就需要很长的base64编码行。