WebPhone parameter encoding

General considerations

The following methods can be used to secure the webphone usage:
-set the loglevel to 1 (with loglevel 5 the password *might be* written in the logs)
-don't hardcode the password if possible (let the users to enter it) or if you must hardcode it then use encryption and/or obfuscation
-restrict the account on the VoIP server (for example if the webphone is used as a support access, then allow to call only your support numbers)
-instead of password, use the MD5 and the realm parameters if possible (and this can also passed in encrypted format to be more secure)
-instead of preconfigured parameters you can also use the javascript api (start, register)
-use https (secure http / TLS)
-for parameter encryption/obfuscation you can use XOR + base64 with your built-in key (ask from Mizutech), prefixed with the "encrypted__3__" string (you can verify your encryption with *this tool* using selecting XOR Base64 Encrypt)
-secure your VoIP server (account limits, rate-limits, balance limits, fraud detection)

Parameter encryption/obfuscation

You can encode any webphone parameter, however the only sensitive parameter for the webphone is the user SIP password. The others (such as the SIP server address) are not worth to encode or they can be found anyway from a network trace.

To encode the parameters you just have to prefix them like encrypted__3__encryptedvalue and use xor+base64 for parameter obfuscation.
Example: password: "encrypted__3__d3691adb28b5591",
The same format can be used to encrypt the string parameters for the API function calls.

Note: only ASCII character set is supported at this moment. This means that your parameter to be encrypted cannot contain Unicode-only characters.

Parameter encryption explained

The key in the public demo is "h39idbfqw7116ghh".

To encrypt any string, you have to XOR it with this key byte by byte.
First char with 'h'

Second char with '3'.
Third char with '9'
Etc

If they parameter to encrypt is longer than the key, then just start again (So when the keypos >= keylen, then set the keypos again to 0).

Once the XOR is ready, just Base64 encode the result and you are ready.

For the decoding, do it in the inverse order: first Base64 decode, then XOR.

Encrypted by default

You can ask Mizutech to build your webphone with already encrypted parameters. This can be done for all static parameters which means that you can't encrypt the SIP username/password this way unless you are using a single hardcoded SIP account (useful for click-to-call or similar services to call an access number).

API

You can use the following API's provided by Mizutech to manually or programmatically encode parameters.

https://mnt.mizu-voip.com/mmexreqvitserverjrsktt/xwencode?wkey=YOURKEY&wstring=YOURSTRING&wversion=3

Replace YOURKEY with your webphone key provided by mizutech.
Replace YOURSTRING with a parameter value. URI encode if contains special characters such as spaces.
The answer is a clear text string with the following format:
OK: Encoded string: xxx

You might also verify your encryption by trying to decrypt using the following API:
https://mnt.mizu-voip.com/mmexreqvitserverjrsktt/xwdecode?wkey=YOURKEY&wstring=YOURSTRING&wversion=3
Replace YOURKEY with your webphone key provided by mizutech.
Replace YOURSTRING with an encrypted parameter value.
The answer is a clear text string with the following format:
OK: Decoded string: xxx

PHP example

Xor+base encryption in PHP looks like this: encrypted__3__<? print base64_encode(stringtoencrypt ^ key); ?>

This is only a simplified example. If the key is smaller than the string to encrypt then you must xor it char by char (and restart with the first char of the key after the last one)

Java example

These examples are written in Java but you should be able to find similar functionalities in your preferred language be it JavaScript, PHP, J2EE, .NET or any other programming environment.

```
public static String strxor(String str, String key) {
      try {
      String result = null;
      byte[] strBuf = str.getBytes();
      byte[] keyBuf = key.getBytes();
      int c = 0;
      int z = keyBuf.length;
      ByteArrayOutputStream baos = new ByteArrayOutputStream(strBuf.length);
      for (int i = 0; i < strBuf.length; i++) {
        byte bS = strBuf[i];
        byte bK = keyBuf[c];
        byte bO = (byte)(bS ^ bK);
        if (c < z - 1) {
          c++;
        } else {
          c = 0;
        }
        baos.write(bO);
      }

        baos.flush();
        result = baos.toString();
        baos.close();
        baos = null;
        return result;
      } catch (Exception e) { Common.PutToDebugLogException(3,"xor",e); }
      return Base64Coder.encodeString(str);
    }
```

Java Base64 implementation example

You can use the built-in base64 or any third-party implementation. Here is an example:

```
public class Base64Coder
{

   // Mapping table from 6-bit nibbles to Base64 characters.
   private static char[] map1 = new char[64];
   static
   {
      int i = 0;
```

```
    for(char c = 'A'; c <= 'Z'; c++)
    {
        map1[i++] = c;
    }
    for(char c = 'a'; c <= 'z'; c++)
    {
        map1[i++] = c;
    }
    for(char c = '0'; c <= '9'; c++)
    {
        map1[i++] = c;
    }
    map1[i++] = '+';
    map1[i++] = '/';
}

// Mapping table from Base64 characters to 6-bit nibbles.
private static byte[] map2 = new byte[128];
static
{
    for(int i = 0; i < map2.length; i++)
    {
        map2[i] = -1;
    }
    for(int i = 0; i < 64; i++)
    {
        map2[map1[i]] = (byte)i;
    }
}

/**
 * Encodes a string into Base64 format.
 * No blanks or line breaks are inserted.
 * @param s  a String to be encoded.
 * @return   A String with the Base64 encoded data.
 */
public static String encodeString(String s)
{
    return new String(encode(s.getBytes()));
}

/**
 * Encodes a byte array into Base64 format.
 * No blanks or line breaks are inserted.
 * @param in  an array containing the data bytes to be encoded.
 * @return    A character array with the Base64 encoded data.
 */
public static char[] encode(byte[] in)
{
    return encode(in, in.length);
}

/**
 * Encodes a byte array into Base64 format.
 * No blanks or line breaks are inserted.
 * @param in   an array containing the data bytes to be encoded.
 * @param iLen number of bytes to process in <code>in</code>.
 * @return     A character array with the Base64 encoded data.
```

```
 */
public static char[] encode(byte[] in, int iLen)
{
   int oDataLen = (iLen * 4 + 2) / 3; // output length without padding
   int oLen = ((iLen + 2) / 3) * 4; // output length including padding
   char[] out = new char[oLen];
   int ip = 0;
   int op = 0;
   while(ip < iLen)
   {
      int i0 = in[ip++] & 0xff;
      int i1 = ip < iLen ? in[ip++] & 0xff : 0;
      int i2 = ip < iLen ? in[ip++] & 0xff : 0;
      int o0 = i0 >>> 2;
      int o1 = ((i0 & 3) << 4) | (i1 >>> 4);
      int o2 = ((i1 & 0xf) << 2) | (i2 >>> 6);
      int o3 = i2 & 0x3F;
      out[op++] = map1[o0];
      out[op++] = map1[o1];
      out[op] = op < oDataLen ? map1[o2] : '=';
      op++;
      out[op] = op < oDataLen ? map1[o3] : '=';
      op++;
   }
   return out;
}

/**
 * Decodes a string from Base64 format.
 * @param s  a Base64 String to be decoded.
 * @return   A String containing the decoded data.
 * @throws   IllegalArgumentException if the input is not valid Base64 encoded data.
 */
public static String decodeString(String s)
{
   return new String(decode(s));
}

/**
 * Decodes a byte array from Base64 format.
 * @param s  a Base64 String to be decoded.
 * @return   An array containing the decoded data bytes.
 * @throws   IllegalArgumentException if the input is not valid Base64 encoded data.
 */
public static byte[] decode(String s)
{
   return decode(s.toCharArray());
}

/**
 * Decodes a byte array from Base64 format.
 * No blanks or line breaks are allowed within the Base64 encoded data.
 * @param in  a character array containing the Base64 encoded data.
 * @return    An array containing the decoded data bytes.
 * @throws    IllegalArgumentException if the input is not valid Base64 encoded data.
 */
public static byte[] decode(char[] in)
{
```

```
        int iLen = in.length;
        if(iLen % 4 != 0)
        {
           throw new IllegalArgumentException("Length of Base64 encoded input string is not a multiple of 4.");
        }
        while(iLen > 0 && in[iLen - 1] == '=')
        {
           iLen--;
        }
        int oLen = (iLen * 3) / 4;
        byte[] out = new byte[oLen];
        int ip = 0;
        int op = 0;
        while(ip < iLen)
        {
           int i0 = in[ip++];
           int i1 = in[ip++];
           int i2 = ip < iLen ? in[ip++] : 'A';
           int i3 = ip < iLen ? in[ip++] : 'A';
           if(i0 > 127 || i1 > 127 || i2 > 127 || i3 > 127)
           {
              throw new IllegalArgumentException("Illegal character in Base64 encoded data.");
           }
           int b0 = map2[i0];
           int b1 = map2[i1];
           int b2 = map2[i2];
           int b3 = map2[i3];
           if(b0 < 0 || b1 < 0 || b2 < 0 || b3 < 0)
           {
              throw new IllegalArgumentException("Illegal character in Base64 encoded data.");
           }
           int o0 = (b0 << 2) | (b1 >>> 4);
           int o1 = ((b1 & 0xf) << 4) | (b2 >>> 2);
           int o2 = ((b2 & 3) << 6) | b3;
           out[op++] = (byte)o0;
           if(op < oLen)
           {
              out[op++] = (byte)o1;
           }
           if(op < oLen)
           {
              out[op++] = (byte)o2;
           }
        }
        return out;
     }

     // Dummy constructor.
     private Base64Coder()
     {
     }

} // end class Base64Coder
```

Third-party service

You can verify your encryption with this tool using selecting XOR Base64 Encrypt.