Yarpchiever, an Encryption File Method Using Binary Rotation and Compress it Using gzip at Once

./Introducing

Saat ini banyak sekali software-software gratis maupun berbayar yang dapat digunakan untuk mengenkripsi file. Akan tetapi tidak sedikit pula file encrypted dari software-software tersebut yang dapat di decrypt dengan mudah. Hal ini dikarenakan pola enkripsi yang digunakan software tersebut telah diketahui, baik melalui debugging maupun dengan menganalisa hasil enkripsinya.

Dengan alasan inilah saya lebih suka menggunakan metode enkripsi dengan membuat pola tersendiri. Karena tentunya selain hanya saya sendiri yang mengetahui pola untuk men-decrypt file tersebut, juga fitur-fitur lain yang bisa kita tambahkan sesuka hati. Seperti pada tulisan kali ini, tentang **Enkripsi File Menggunakan Metode Binary Rotation.** Akan tetapi seperti yang kita ketahui, pada umumnya file hasil enkripsi biasanya selalu lebih besar ukurannya dibandingkan dengan file aslinya. Oleh sebab itu, tools yang akan kita buat berikut ini akan kita tambahkan fitur sekaligus untuk **mengkompresi file** tersebut.

Binary Rotation

Sedikit gambaran mengenai Binary Rotation (istilah buatan sendiri :p), konsepnya agak mirip dengan ROT13, hanya saja rotasi dilakukan pada level Binary, dengan jumlah rotasi berdasarkan *Token key* yang digenerate secara random dari aplikasi yang nantinya *Token key* ini akan digunakan kembali untuk men-decrypt.

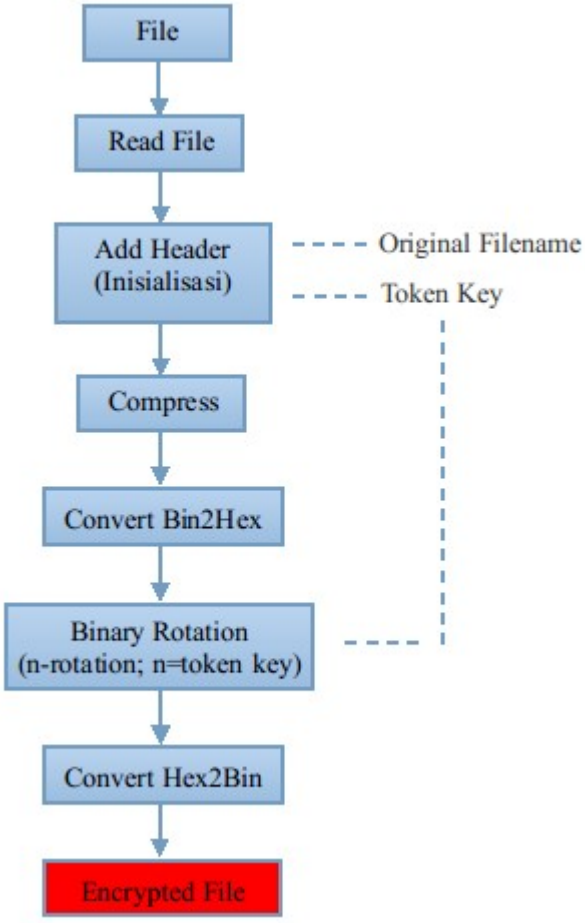
Gzip Compression

Berikutnya adalah fitur *kompresi* dengan memanfaatkan library *Gzip.* Hal ini dimaksudkan agar dalam pengembangannya fitur ini kompatibel dengan bahasa pemrograman apapun.

./Proof Of Concept

Sebenarnya alur dari metode enkripsi ini cukup sederhana, seperti digambarkan pada alur berikut ini:

Alur Enkripsi Yarpchiever



Karena basic pengetahuan bahasa pemrograman saya masih sebatas PHP, sehingga implementasi enkripsi file diatas masih menggunakan bahasa pemrograman PHP-CLI (Semoga dalam pengembangannya bisa dibuat dalam bahasa pemrograman bahasa C, biar bisa di compile. :p).

Penjelasan alur diatas adalah sebagai berikut:

- Add Header

Pada tahap ini, file yang akan dienkripsi ditambahkan header untuk inisialisasi saat file akan di decrypt nantinya. **Original Filename** digunakan untuk menyimpan nama file asli jika pada saat dekripsi tidak diberikan opsi untuk penamaan output file hasil dekripsi. Jika parameter nama baru tidak diberikan, maka secara default, nama file asli sebelum di *encrypt* yang akan digunakan.

Sedangkan pada Token Key, terdapat *Hash* jumlah rotasi. *Hash* ini digenerate menggunakan fungsi **fakemd5_encode()**

[code]

```
function fakemd5_encode($str,$bool=false){
  $len = strlen($str);
  $hex = array(1=>1,2,3,4,5,6,7,8,9,"a","b","c","d","e","f");
  if($len>15){
      return false;
```

```
        } else {
            $hash = NULL;
            $rand = generateChars(32,$bool);
            $obj = strrev($str);
            $used = $len * 2;
            $cover = substr($rand,$used,31-$used);
            for($i=0;$i<$len;$i++){
                $hash .= substr($rand,$i * 2 ,1).substr($obj,$i,1);
            }
            return $hash.$cover.$hex[$len];
        }
    }
```

[/code]

- Compress File

Pada tahap ini, file dikompress menggunakan gzip. *(Source code selengkapnya di lampiran)*

- Convert Bin2Hex

Setelah file dikompres, kemudian binary file diparsing dalam bentuk String dengan format *Hexa,* untuk kemudian dirotasi.

- Binary Rotation

Seperti pada metode enkripsi ROT13, raw data file digeser per bitnya kearah kanan sebanyak **Token Key** yang dihasilkan.

Ex:

abcdef --→ bcdefa (rotasi 1x)
abcdef --→ fabcde (rotasi 5x)

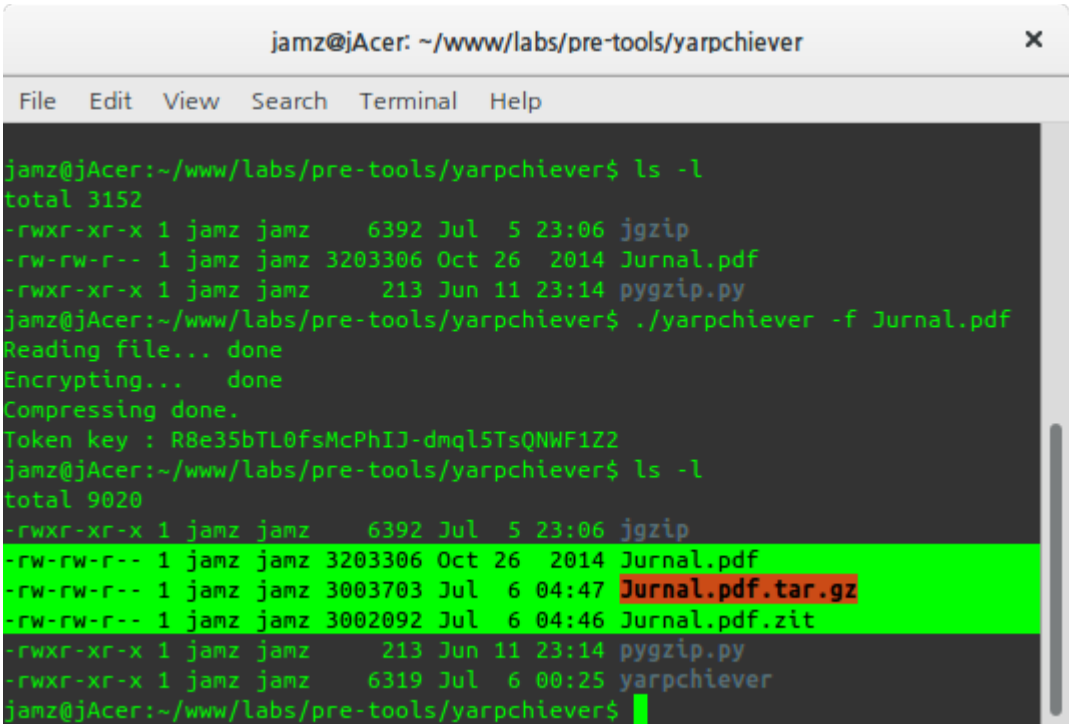
- Convert Hex2Bin

Setelah raw data dirotasi, berikutnya adalah mengubah kembali format hexa tersebut menjadi format binary, kemudian simpan kedalam file dengan metode gzip agar nantinya saat di decrypt, cara pembacaan file tetap seragam saat dibaca kembali menggunakan bahasa pemrograman apapun.

Berikut ini contoh penggunaan yarpchiever untuk mengenkripsi file. Sebagai contoh pada gambar adalah enkripsi file Jurnal.pdf dengan ukuran 3.2 MB. Setelah dienkripsi plus kompresi menggunakan yarpchiever ukuran file menjadi 3.0 MB. Pada perintah "*ls"* file berikutnya, saya coba bandingkan kompresi file Jurnal.pdf menjadi *tar.gz,* dengan hasil sebagai berikut:

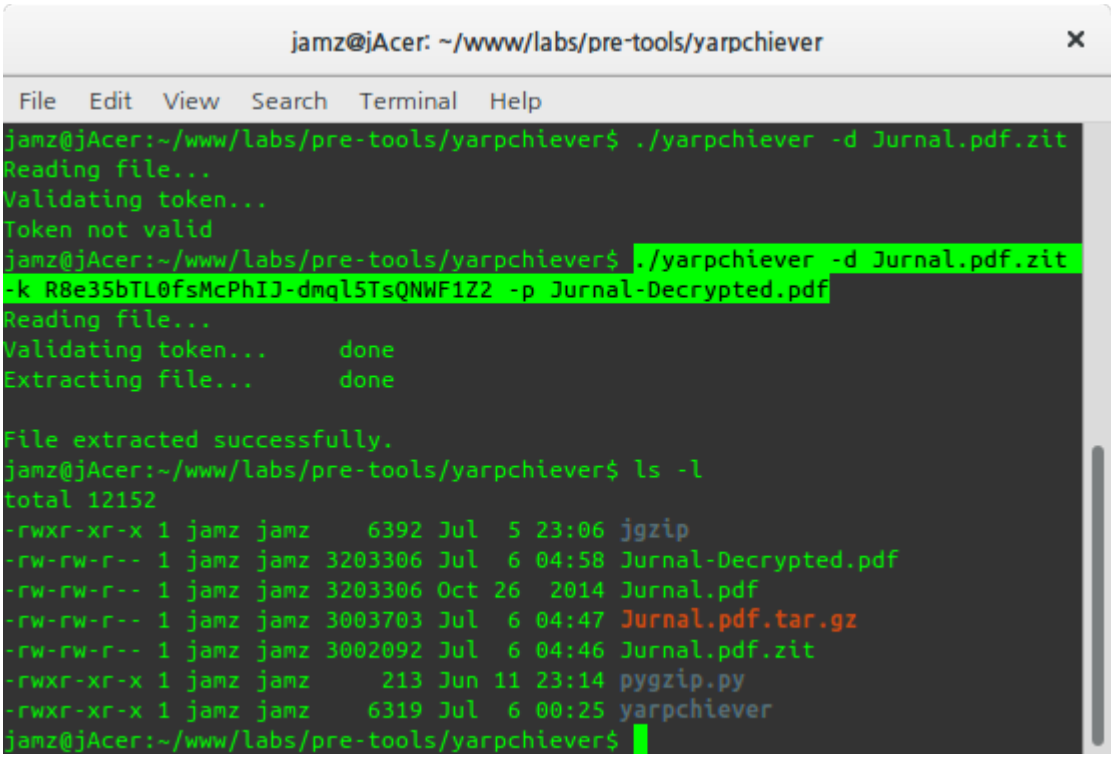
Jurnal.pdf	3203306 KB
Jurnal.pdf.tar.gz	3003703 KB
Jurnal.pdf.zit	3002092 KB

Tampak diatas ukuran hasil kompresi menggunakan yarpchiever lebih kecil daripada tar.gz :)



Perhatikan pada gambar diatas, setelah melakukan enkripsi, kita akan mendapatkan sebuah kode **Token Key : R8e35bTL0fsMcPhIJ-dmql5TsQNWF1Z2.** Token Key inilah yang akan kita gunakan untuk mendecrypt file Jurnal.pdf.zit

Proses Decrypting File



Tampak juga jika Token Key tidak diinputkan, maka file tidak akan di decrypt.

Agar lebih menarik, Yarpchiever disertai dengan beberapa parameter khusus masing-masing untuk enkripsi dan dekripsi.

[code]
```
Usage:
$ ./yarpchiever [-d|-f] FILENAME [parameter]
Parameter:
Enkripsi
      [wajib]:
            -f file yang dienkripsi
      [optional]
            -o nama file Output compressed file
            -w output token key ke dalam file .token
            -r delete original file setelah enkripsi
Dekripsi
      [wajib]
            -d file yang akan didecrypt
            [-k|-u] Token key untuk decrypt file. "-k" untuk input token langsung
dilayar. Dan parameter "-u" input token dari file ".token".
      [optional]
            -p nama file Output file decrypted (default: set to original file name)

Example:
      Compress : ./yarpchiever -f file1 -o file1.zit
      Decompress: ./yarpchiever -d file1.zit -p file2 -k [tokenkey]
```
[/code]

./Summary

Setiap aplikasi tak akan lepas dengan yang namanya kelebihan dan kekurangan. Yarpchiever merupakan tools sederhana yang awalnya hanya bersifat *"Private Use" ,* tapi tidak ada salahnya jika dipublish, ada yang berminat untuk mengembangkan, menulis kembali program ini menggunakan bahasa pemrograman yang lebih powerfull tetap dengan menggunakan Logika Yarpchiever.

Berikut ini beberapa kelebihan dan kekurangan dari Yarpchiever yang mungkin bisa dijadikan sebagai bahan diskusi:

Kelebihan :

- Relative lebih aman karena pola enkripsi yang tidak (belum) diketahui oleh publik
- Enkripsi file sekaligus mengkompress hasil file yang terenkripsi
- User tidak perlu membuat password, tetapi Kunci akan digenerate oleh aplikasi.
- 1 file yang sama, jika di enkripsi 2 kali, masing-masing akan menghasilkan file dan men-*generate* Token Key yang berbeda. Dan masing-masing Token hanya bisa digunakan untuk men-decrypt masing-masing file enkripsi
- Ukuran file lebih kecil dari hasil kompresi biasa menggunakan tar.gz

- Cross Platform
- Open Source, sehingga fitur-fitur bisa ditambahkan oleh pengembang

Kekurangan :

- Karena ditulis menggunakan bahasa pemrograman yang tidak bisa dikompile, sehingga alur enkripsi mudah untuk di baca dan di debug.

./Regards

Yarp, semoga tulisan ini bisa bermanfaat dalam hal mengamankan file-file penting. Jika Yarpchiever tidak bisa disebut sebagai Tools Enkripsi yang baik, setidaknya dengan **Tidak Menggunakan Software-Software Enkripsi File yang Umum ,** file-file enkripsi kita tidak bisa di decrypt dengan mudah menggunakan cara-cara yang umum pula.

K4pT3N – *6 Juli 2015*

./Lampiran {SourceCode:Full}

```
#!/usr/bin/php -q
<?php
	/* ============================================================== *\
	 *	Yarpchiever v1.0
	 * ==============================================================
	 *	Author		: K4pT3N
	 *	URL		: http://codernate.org | http://explorecrew.org
	 *	Contact		: kapten.mozac@gmail.com
	 *
	\* ============================================================== */

	class yarpchiever {
		function __construct(){
			($_SERVER['argc']==1)?$this->help():$this->param_main();
		}
		function param_main(){
			$rparam=$_SERVER['argv'];
			(in_array('-d',$rparam))?$this->decompress_procedure():$this-
>compress_procedure();
		}
		function param_value($idx){
			return $_SERVER['argv'][$idx+1];
		}
		function param_key_index($param){
			return array_search($param,$_SERVER['argv']);
		}
		function compress_procedure(){
			$filename=$this->param_value($this->param_key_index('-f'));
			$this->param_f($filename);
		}
		function decompress_procedure(){
			$filename=$this->param_value($this->param_key_index('-d'));
			$this->param_d($filename);
		}
		function param_f($filename){
			echo "Reading file...";
			if(file_exists($filename)){
				$f=file_get_contents($filename);
				$content='yarp|'.$filename.'|'.$f;
				$zip=gzcompress($content,9);
				echo "\tdone\n";
				$this->encrypt($filename,$zip);
			} else {
				echo "File not found.\nExit.\n";
			}
		}
		function param_o($filename,$contents){
			$key=$this->param_key_index('-o');
```

```
        $output=($key=="")?$filename.'.zit':$this->param_value($key);

        $fp=gzopen($output,"w9");
        $packed=hex2bin($contents);
        gzwrite($fp,$packed);
        gzclose($fp);
        if($this->param_key_index('-r')>0) unlink($filename);
}
function param_p($filename,$contents){
        $p=$this->param_key_index('-p');
        $output=($p>0)?$this->param_value($p):$filename;
        $fp=fopen($output,"wb");
        fwrite($fp,$contents);
        fclose($fp);
        echo "\tdone\n";
        echo "\nFile extracted successfully.\n";
}
function param_d($filename){
        if(file_exists($filename)){
                echo "Reading file...\n";
                echo "Validating token...";
                $token=$this->is_token();
                if($token){
                        echo "\tdone\n";
                        echo "Extracting file...";
                        $f=file_get_contents($filename);
                        $rest=substr($f,-4);
                        $fpsize=end(unpack("V",$rest));
                        $fp = gzopen($filename, "rb");
                        $fread = gzread($fp, $fpsize);
                        gzclose($fp);

                        $hex=bin2hex($fread);
                        $rotated=yarpRotate($hex,$token,-1);
                        $bin=hex2bin($rotated);
                        $ungzip=gzuncompress($bin);
                        $header=explode("|",$ungzip,3);
                        if($header[0]!=="yarp"){
                                echo "\nFalse token\n";
                        } else {
                                $this->param_p($header[1],$header[2]);
                        }
                } else {
                        echo "\nToken not valid\n";
                }
        } else {
                echo "File not found.\nExit.\n";
        }
}
function is_token(){
```

```
                $k=$this->param_key_index('-k');
                if($k>0){
                        return fakemd5_decode($this->param_value($k));
                } else {
                        $u=$this->param_key_index('-u');
                        if($u>0){
                                if(file_exists($this->param_value($u))){
                                        $fp=file_get_contents($this->param_value($u));
                                        return fakemd5_decode($fp);
                                } else {
                                        return false;
                                }
                        } else {
                                return false;
                        }
                }
        }
        function encrypt($filename,$contents){
                echo "Encrypting...";
                $hex=bin2hex($contents);
                $salt=generateSalt($hex);
                $rot=fakemd5_decode($salt);
                $rotated=yarpRotate($hex,$rot);
                echo "\tdone\n";
                echo "Compressing done.\n";
                if($this->param_key_index('-w')>0){
                        $fp=fopen(".token","w");
                        fwrite($fp,$salt);
                        fclose($fp);
                        echo "Token key saved to .token\n";
                } else {
                        echo "Token key : ".$salt."\n";
                }
                $this->param_o($filename,$rotated);
        }
        function help(){
                echo "Usage:\n";
                echo "$ ./yarpchiever [-d|-f] FILENAME [parameter]\n";
                echo "Parameter:\n";
                echo "Compress \n\t[required]:\n\t\t-f filenya\n\t[optional]\n\t\t-o Output compressed file\n\t\t-w output token key to file\n\t\t-r delete original file\nDecompress\n\t[required]\n\t\t-d file to decompress\n\t\t[-k|-u] Token's key to decompress compressed file. \"-k\" input token manually. \"-u\" input token from existing file \".token\".\n\t[optional]\n\t\t-p Output decompressed file (default: set to original file name)\n\n";
                echo "Example:\n\tCompress : ./yarpchiever -f file.txt -o file.txt.zit\n";
                echo "\tDecompress: ./yarpchiever -d file.txt.zit -p file1.txt -k [tokenkey]\n\n";
        }
}
$y=new yarpchiever;
```

```
    function generateSalt($str){
            return fakemd5_encode(rand(1,1000),true);
    }
    function fakemd5_encode($str,$bool=false){
  $len = strlen($str);
  $hex = array(1=>1,2,3,4,5,6,7,8,9,"a","b","c","d","e","f");
  if($len>15){
    return false;
  } else {
    $hash = NULL;
    $rand = generateChars(32,$bool);
    $obj = strrev($str);
    $used = $len * 2;
    $cover = substr($rand,$used,31-$used);
    for($i=0;$i<$len;$i++){
      $hash .= substr($rand,$i * 2 ,1).substr($obj,$i,1);
    }
    return $hash.$cover.$hex[$len];
  }
}
function generateChars($limit,$specialchars=false,$type='all'){
        $hexa=NULL;
        $chars = ($specialchars===false)?null:"^-+";
        $chrup='ABCDEFGHIJKLMNOPQRSTUVWXYZ';
        $chrlow='abcdefghijklmnopqrstuvqxyz';
        $chrnum='0123456789';
        switch($type){
                case 'up'       : $chr=$chrup.$chars; break;
                case 'low'      : $chr=$chrlow.$chars; break;
                case 'num'      : $chr=$chrnum.$chars; break;
                case 'uplow': $chr=$chrup.$chrlow.$chars; break;
                case 'upnum': $chr=$chrup.$chrnum.$chars; break;
                case 'lownum': $chr=$chrlow.$chrnum.$chars; break;
                default         : $chr=$chrup.$chrlow.$chrnum.$chars; break;
        }
        for($i=1;$i<$limit+1;$i++){
                $rIdx = rand(1,strlen($chr));
                $hexa .=substr($chr,$rIdx,1);
        }
        return $hexa;
    }
function fakemd5_decode($str){
  $hex = array(1=>1,2,3,4,5,6,7,8,9,"a","b","c","d","e","f");
  $len = array_search(substr($str,-1,1),$hex);
  $hash = NULL;
  for($i=0;$i<$len;$i++){
    $hash .= substr($str,$i * 2+1,1);
  }
  return strrev($hash);
}
```

```
function yarpRotate($bin,$n,$rot=1){
	$n=$n*$rot;
	$output=null;
	for($i=0;$i<strlen($bin);$i++){
		$x=$i+$n;
		if($x>strlen($bin)-1) $x=$x-strlen($bin);
		$output .= substr($bin,$x,1);
	}
	return $output;
}
```