Engenharia reversa de código malicioso

Italo Valcy[1,2]
[1]CERT.Bahia – PoP-BA/RNP
[2]Universidade Federal da Bahia

italo@pop-ba.rnp.br

III EnSI CERT.Bahia, 29/Nov/2013

Engenharia reversa

O que é e pra que serve

- Processo de descoberta do funcionamento de um programa de computador sem ter acesso ao código fonte.
- Pode ser simplesmente executá-lo ou mesmo analisar suas instruções de máquina
- Deseja-se saber:
  - O que o programa faz?
  - Qual parte do programa faz acesso a rede?
  - Quais parâmetros?
  - Como é ativado?

Engenharia reversa

O que é e pra que serve

- Exemplos de aplicação:
  - Desenvolvimento de drivers
  - Desenvolvimento de software livre a partir de proprietário
  - Entendimento de protocolos para compatibilidade
  - Documentação de código legado
  - Análise de código malicioso

Tipos de programas maliciosos

- Vírus
- Trojan
- Worm
- Spyware
- Rootkit

Formas de ataque

- Vetores de ataque de baixo nível:
  - Estouro de buffers
  - Vulnerabilidades formato de string
- Vetores de ataque de alto nível:
  - Exploração de vulnerabilidades em aplicações
  - Falhas em configurações e proteções
- Ataques de mais alto nível:
  - Engenharia social
  - E-mails em massa / phishing

Classificação código malicioso

- Código constante: executa de forma constante
  - Ex: loop de coleta de e-mails, scan de rede
- Código reativo: executado em resposta a evento específico
  - Ex: Quando usuário acessa determinada página
- Código dormente: executa em data determinada
  - Ex: ataque DDoS coordenado

Ambiente de análise

- Análise “ao vivo”
  - Executar o programa e monitorá-lo
  - Permite identificar rapidamente o objetivo do programa
  - Difícil detectar a forma como o programa executa
- Engenharia reversa
  - Análise do executável do programa
  - Permite entender o funcionamento, descobrir código dormente ou reativo
  - Alta complexidade

Ambiente de análise

- Ambiente instalado com finalidade de análise
  - Forma mais segura
  - Gasta muito tempo e recurso
  - Pode-se usar hardware/software de clone e recuperação de disco
- Ambiente virtual
  - Pode simular um ambiente completo
  - VMWare, Xen, KVM, etc.
  - Facilita recuperação do estado inicial
  - Pode ser detectado pelo código malicioso

Ferramentas

- Máquinas virtuais
- Ferramentas de análise dinâmica
- Debuggers
- Decompiladores
- Disassemblers
- Ferramentas de manipulação de arquivos executáveis

Debuggers

- Informações sobre o estado da CPU
- Execução passo-a-passo
- Pontos de parada (break-points)
- Permite avançar e retroceder
- Visualização e manipulação de memória e registradores
- Visualização de threads

Debuggers

Debuggers para Windows:

- Microsoft WinDBG
- OllyDBG
- IDA Pro
- PyDBG

Debuggers para Linux:

- GDB
- DBX
- Valgrind (memória)

Decompilador

- Tentar traduzir binário em linguagem de alto nível
  - Geralmente apresenta muitas falhas
  - Útil para deixar o disassembly mais legível
- Exemplo:
  - REC e REC Studio
  - Desquirr
  - Boomerang
  - Hex-Rays Decompiler

Disassembler

- Ferramenta de análise estática, transforma bytes em linguagem assembly
- Geralmente os debuggers fazem isso
- Desafio: diferenciar código de dados

  - IDA Pro
  - OllyDBG
  - Fenris
  - PE Browser
  - Objdump
  - Ndisasm

Analisadores de arquivos

- Formato padronizado para o sistema operacional
  - Local do código, bibliotecas, mapeamento memória
- Formatos de arquivos executáveis
  - Portable Executable PE (Windows, DOS)
  - ELF (Linux, Unix)
  - ABI Mach-O (MacOS X)

Analisadores de arquivos

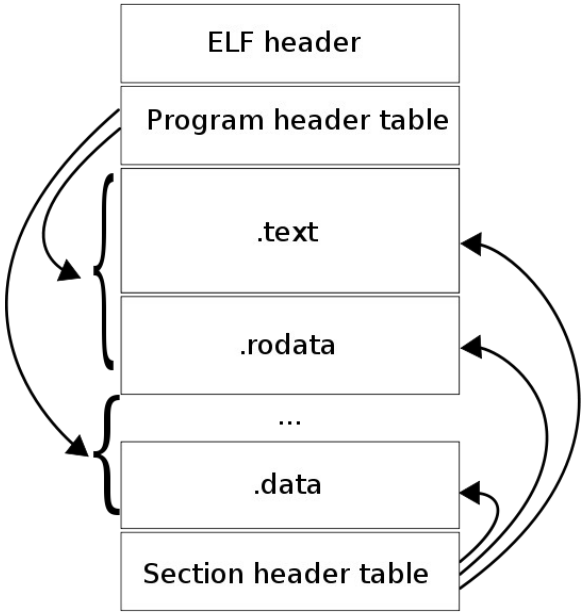
Portable Executable

- Por que “portável”?
  - Suporta x86 32 bits e 64 bits
  - Suporta arquitetura MIPS, PowerPC, ARM, DEC
- Extensão .exe ou .dll
- Seções:
  - .text
  - .data
  - .idata, .edata

Analisadores de arquivos

ELF

- Executáveis, shared libs, código objeto
- Utilizado em Linux, FreeBSD, Solaris, PlayStation, Android, etc.
- Diversas ferramentas de manipulação:
  - Binutils
  - Elfutils



Analisadores de arquivos

- Ferramentas para Linux:
  - file
  - strings
  - nm
  - grep
  - stat

Assembly

- Assembly: necessário para Engenharia reversa
  - x86: CPU, Memória, registradores e disco
  - Recupera, decodifica, executa
  - Aplicação: conjunto de instruções assembly

```
int main (){
    printf("Hello World");
}
```

```
HelloWorld proc near
var_8= dword ptr -8
var_4= dword ptr -4
push   ebp
mov    ebp, esp
sub    esp, 8
and    esp, 0FFFFFFF0h
mov    eax, 0
add    eax, 0Fh
add    eax, 0Fh
shr    eax, 4
shl    eax, 4
mov    [ebp+var_4], eax
mov    eax, [ebp+var_4]
call   sub_401088
call   __main
mov    [esp+8+var_8], offset aHelloWorld; "Hello World"
call   printf
leave
retn
HelloWorld endp
```

Assembly

- Registradores
  - EAX, EBX, ECX, EDX, ESI, EDI, ESP, etc.
- Stack
  - PUSH / POP
- Instruções:
  - INC, DEC, ADD, SUB, MUL, DIV
  - MOV, LEA
  - CALL / RET, ENTER / LEAVE
  - CMP, TEST
  - JMP, JZ, JNZ, JG, JL, JGE, etc.
  - AND, OR, XOR, etc.

Assembly

```
if (var > 128)
   var = 128;
```

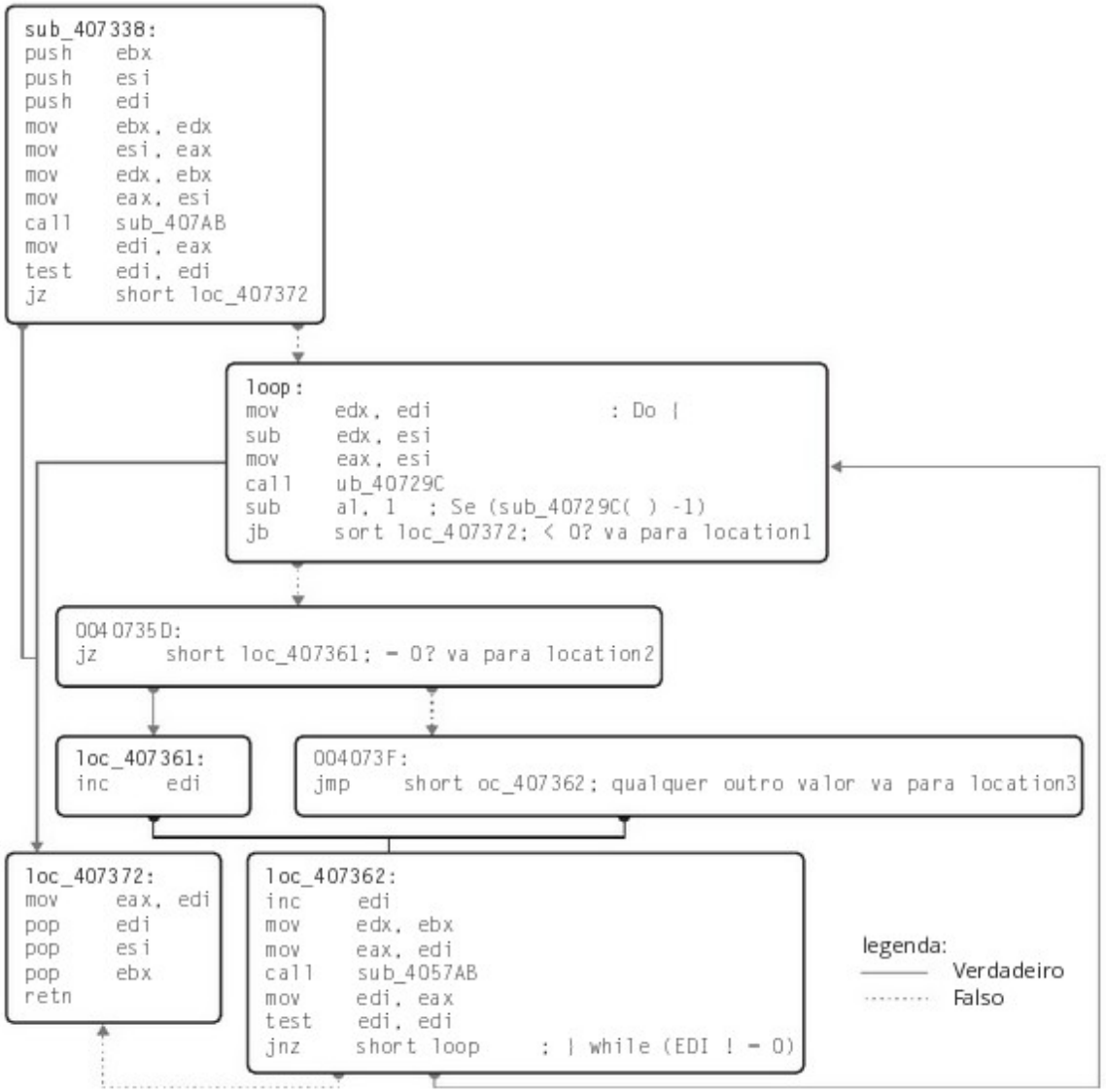
```
mov eax, [ebp+8]
cmp eax, 0x80
jbe skip
mov eax, 0x80
skip:
```

```
for (i = 0; i < 100; i++)
   do_something();
```

```
xor eax, eax
start:
cmp eax, 0x64
jge exit
call do_something
inc eax
jmp start
exit:
```

Assembly

Exemplo de fluxo gerado pelo IDA Pro



Truques anti-engenharia reversa

- Detecção de debugger
- Execução de código antes do Entry Point
- Remoção de seções / símbolos (strip)
- Códigos não alinhados
- Modificações nos cabeçalhos PE / ELF / etc.
- Detecção de máquinas virtuais
- Compactação ou criptografia do binário
- Arquivo mutável

Exemplo de análise

- Exemplo: desafio de segurança do Dragon Research Group:
  - https://dragonresearchgroup.org/challenges/
- Desafio (ago/2013): What is in this zip file? Without altering the original, underlying file, can you get it to show you something mythic?
  - https://dragonresearchgroup.org/challenges/201308/201308.zip
  - Solução: https://dragonresearchgroup.org/challenges/201308/drg.txt

DRG Challenge Ago/2013

- Primeiro passo:

```
italo@oxente /t/drg201308> ls
201308.zip
italo@oxente /t/drg201308> file 201308.zip
201308.zip: Zip archive data, at least v2.0 to extract
italo@oxente /t/drg201308> unzip 201308.zip
Archive:  201308.zip
[201308.zip] 201308 password:
password incorrect--reenter:
password incorrect--reenter:
   skipping: 201308                  incorrect password
italo@oxente /t/drg201308> fcrackzip -D -p /tmp/pw-word-list.txt 201308.zip
possible pw found: 123456 ()
italo@oxente /t/drg201308> unzip 201308.zip
Archive:  201308.zip
[201308.zip] 201308 password:
  inflating: 201308
italo@oxente /t/drg201308>
```

DRG Challenge Ago/2013

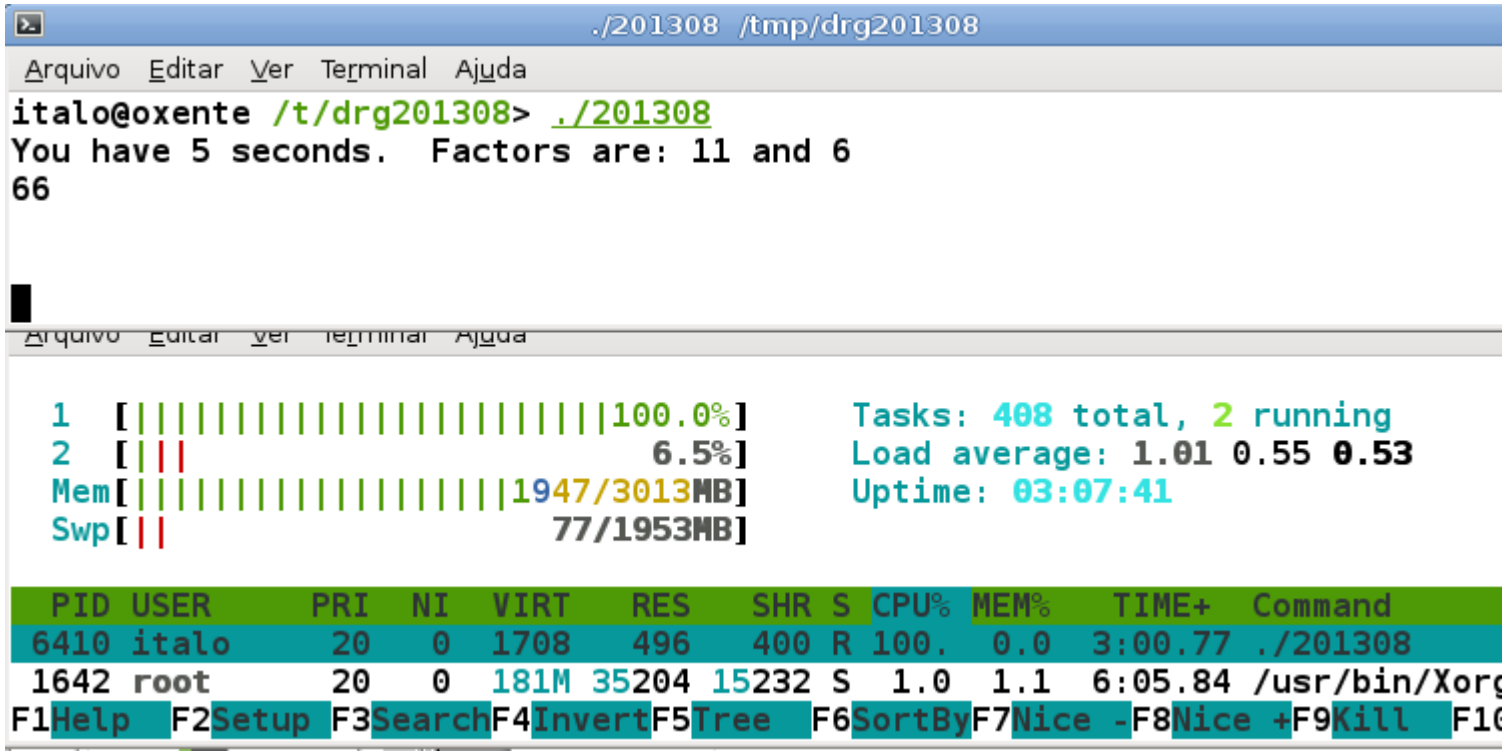
- Segundo passo:

```
italo@oxente /t/drg201308> ls
201308*  201308.zip
italo@oxente /t/drg201308> file 201308
201308: ELF 32-bit LSB executable, Intel 80386, version 1 (GNU/Linux), staticall
y linked, stripped
italo@oxente /t/drg201308> objdump -d 201308

201308:     file format elf32-i386

italo@oxente /t/drg201308>
```

- Fail! :-(
- E se executarmos?

DRG Challenge Ago/2013

- Terceiro passo: execução



- Fail! Loop infinito (<== técnica anti-eng reversa)

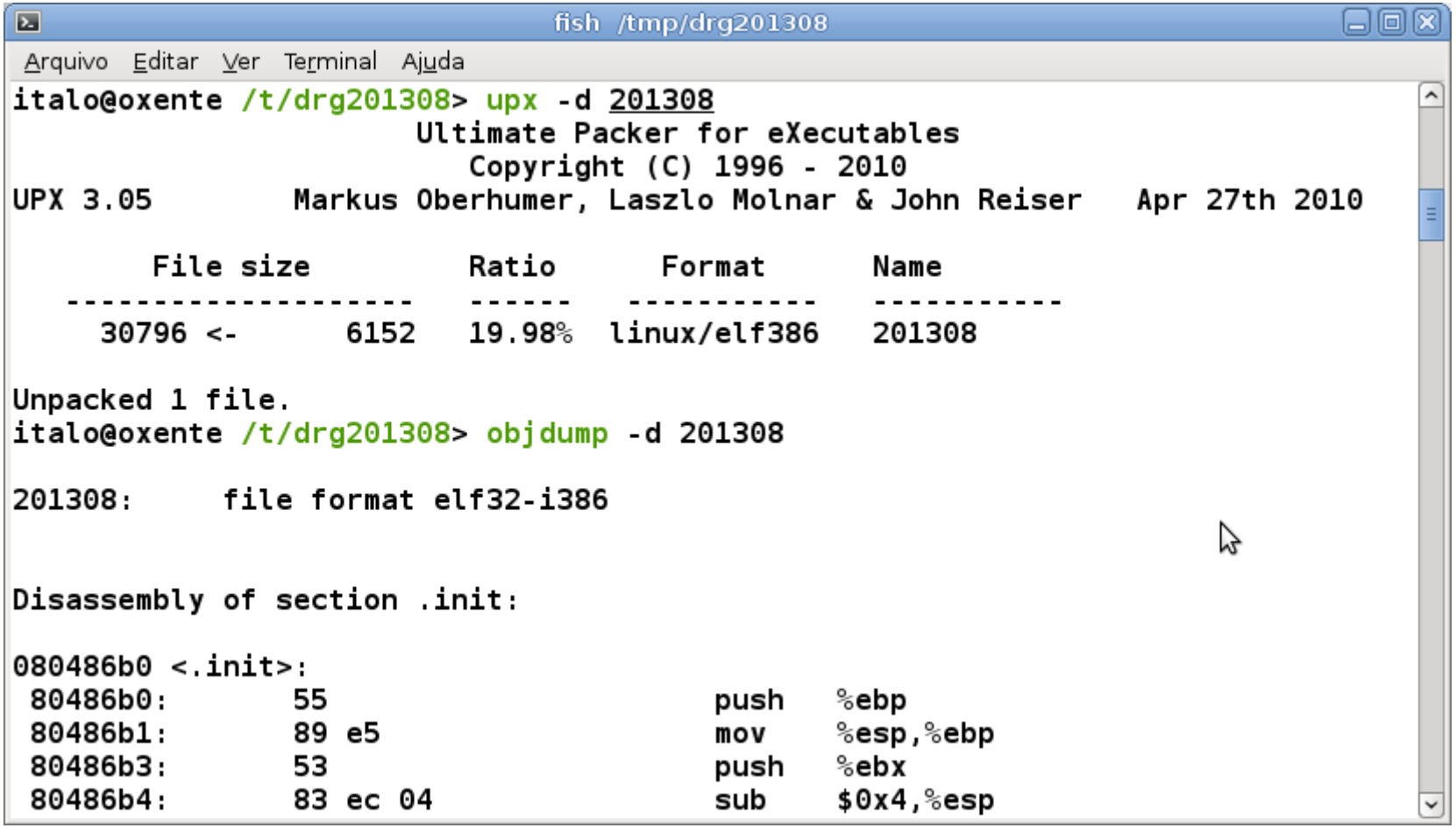
DRG Challenge Ago/2013

- Passo N:
  - Teste com strace => Loop
  - Disassembler => Fail
  - strings: hummm ;)

```
italo@oxente /t/drg201308> strings 201308
UPX!
/lib
nux.so.2
...
$Info: This file is packed with the UPX executable packer http://upx.sf.net $
$Id: UPX 3.05 Copyright (C) 1996-2010 the UPX Team. All Rights Reserved. $
PROT_EXEC|PROT_WRITE failed.
...
B   (hFso]
_fi~O
DjL1
(ox
B`zY
UPX!
```

DRG Challenge Ago/2013

- Passo N+1:
  - UPX decoder + GDB



DRG Challenge Ago/2013

- Técnicas anti – engenharia reversa:
  - Checagem do File Descriptor de arquivos abertos
  - Checagem fork() + ptrace detect
  - Tempo de execução
  - Subrotinas de distração

DRG Challenge Ago/2013

- Técnicas anti – engenharia reversa:

```
#include <stdio.h>
#include <unistd.h>

void detect_gdb(void) __attribute__((constructor));

void detect_gdb(void) {
   FILE *fd = fopen("/tmp", "r");
   if (fileno(fd) > 5) {
      printf("fuck you gdb!\n");
      _exit(1);
   }
   fclose(fd);
}

int main(void) {
   printf("do stuff outside gdb\n");
   return 0;
}
```

DRG Challenge Ago/2013

- Técnicas anti – engenharia reversa:

```
void anti_ptrace(void)
{
    pid_t child;

    if(getenv("LD_PRELOAD"))
        while(1);

    child = fork();
    if (child)
        wait(NULL);
    else
    {
        pid_t parent = getppid();

        if (ptrace(PTRACE_ATTACH, parent, 0, 0) < 0)
            while(1);

        sleep(1);
        ptrace(PTRACE_DETACH, parent, 0, 0);
        exit(0);
    }
}
```

DRG Challenge Ago/2013

- O que o programa faz:

```
804e9b5:   e8 d6 9d ff ff          call   8048790 <uname@plt>
804e9ba:   c7 44 24 04 00 00 00    movl   $0x0,0x4(%esp)
804e9c1:   00
804e9c2:   89 1c 24                mov    %ebx,(%esp)
804e9c5:   e8 e6 9d ff ff          call   80487b0 <gettimeofday@plt>
804e9ca:   69 94 24 b0 01 00 00    imul   $0xf4240,0x1b0(%esp),%edx
804e9d1:   40 42 0f 00
804e9d5:   69 84 24 b8 01 00 00    imul   $0xfff0bdc0,0x1b8(%esp),%eax
804e9dc:   c0 bd f0 ff
804e9e0:   8d 04 02                lea    (%edx,%eax,1),%eax
804e9e3:   03 84 24 b4 01 00 00    add    0x1b4(%esp),%eax
804e9ea:   2b 84 24 bc 01 00 00    sub    0x1bc(%esp),%eax
804e9f1:   3d c8 5e 4c 00          cmp    $0x4c5ec8,%eax
804e9f6:   7f 12                   jg     804ea0a <exit@plt+0x618a>
804e9f8:   89 7c 24 04             mov    %edi,0x4(%esp)
```

DRG Challenge Ago/2013

- Script helper para solução:

```
while read LINE; do
    NUM=$(echo "$LINE" | \
        perl -ne 'if ($_ = /Factors are: (\d+) and (\d+)/) { print $1*$2}')
    if [ -n "$NUM" ]; then
        echo "drg$NUM" > /etc/hostname
        /etc/init.d/hostname.sh
    fi
done
```

DRG Challenge Ago/2013

- Em resumo:
  - Quebra de senha do arquivo zip
  - Desempacotar com UPX
  - Navegar pelo debugger fugindo das armadilhas
  - Encontrar a operação correta de multiplicação e comparação
  - Notar a string “drg%d” nas strings do binário
  - Executar o arquivo original, mudar o hostname e obter o ASCII mítico!

Conclusões

- Engenharia reversa é uma área muito interessante e desafiadora
- Exige conhecimentos em diversas áreas
- O processo de análise precisa ser bem definido, pois exige dedicação e tempo
  - Feito por organizações especializadas
- Pode revelar informações e comportamentos bem interessantes na rede
  - Proteção na rede da organização
  - Detecção de ataques não conhecidos

Dúvidas?

Obrigado!!!
;-)

Perguntas?

Italo Valcy <italo@pop-ba.rnp.br>