Linx Impulse

RELATÓRIO TÉCNICO DO
Processo seletivo para Cloud Engineering

Nome: **Alessandro Elias**

E-mail: ale.elias2011gmail.com

fone: +55 42 99888 8962

Curitba

02 de fevereiro de 2020

Sumário

1 DESAFIO - Parte 1

Estamos enviando pelos links abaixo alguns binários executáveis (ELF 64-bit LSB) que realizam tarefas bem simples, que podem ou não ser úteis. O exercício é que você descubra o que esses binários fazem, utilizando as ferramentas que julgar mais adequadas. Como resposta, esperamos que você nos diga o que você acha que eles fazem e quais foram as ferramentas usadas para isso, bem como uma linha geral do seu ra- ciocínio para chegar às conclusões.

Binários:

cc9621

Primeiramente utilizei a ferramenta *objdump*, e posteriormente para afirmar a conjectura, o *gdb* (ferramentas Unix). *Objdump* oferece algumas análises do cabeçalho e instruções, mnemônicos da arquitetura em que foi compilado o executável, em nosso caso, o formato Executable and Linkable Format (ELF). Com esta ferramenta, extrai (através de engenharia reversa) apenas a seção que nos interessa, a main. No qual representa o ponto de entrada de execução da primeira instrução do programa. O *gdb* foi utilizado para executar o debug, do passo a passo do programa, no qual foi transcrito para um programa na Linguagem C, como mostra o fonte em **??**.

Observações (a versão das ferramentas utilizadas nesta análise são):

```
GNU objdump (GNU Binutils) 2.33.1
Copyright (C) 2019 Free Software Foundation, Inc.
This program is free software; you may redistribute it under the terms of
the GNU General Public License version 3 or (at your option) any later version.
This program has absolutely no warranty.
```

```
GNU gdb (GDB) 8.3.1
Copyright (C) 2019 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.
```

Abaixo mostra a tabela dinâmica de símbolos obtida com o comando:
objdump -T cc9621

```
 1: cc9621:     file format elf64-x86-64


DYNAMIC SYMBOL TABLE:
 2: 0000000000000000      DF *UND* 0000000000000000  GLIBC_2.2.5 getenv
 3: 0000000000000000      DF *UND* 0000000000000000  GLIBC_2.2.5 fclose
 4: 0000000000000000      DF *UND* 0000000000000000  GLIBC_2.4   __stack_chk_fail
```

```
 5: 0000000000000000      DF *UND* 0000000000000000  GLIBC_2.2.5 fputs
 6: 0000000000000000      DF *UND* 0000000000000000  GLIBC_2.2.5 __libc_start_main
 7: 0000000000000000  w   D  *UND* 0000000000000000              __gmon_start__
 8: 0000000000000000      DF *UND* 0000000000000000  GLIBC_2.2.5 fopen
 9: 0000000000000000      DF *UND* 0000000000000000  GLIBC_2.7   __isoc99_scanf
10: 0000000000000000      DF *UND* 0000000000000000  GLIBC_2.2.5 strcat
```

Podemos observar na linha 1 que o programa foi compilado para arquitetura x86 de 64bits, logo veremos apenas mnemônicos x86. Das linhas 2 à 10 observamos as chamadas a libs dinâmicas.

Para obter uma visão transversal do que o programa faz, foi feito a desmontagem do programa, utilizando o comando:

objdump -d –no-show-raw-insn cc9621

```
000000000040072d <main>:
  40072d:       push   %rbp
  40072e:       mov    %rsp,%rbp
  400731:       push   %rbx
5 400732:       sub    $0x1f8,%rsp
  400739:       mov    %fs:0x28,%rax
  400742:       mov    %rax,-0x18(%rbp)
  400746:       xor    %eax,%eax
  /*
10  * The instruction below assign the string "/tmp/".
  * Note the optimization of the compiler result in a
  * direct assignment, no memoey access.
  * (Little endian) 0x2f=/ 0x74=t 0x6d=m 0x70=p 0x2f=/
  */
15 400748:       movabs $0x2f706d742f,%rax
  400752:       mov    %rax,-0x1f0(%rbp)
  400759:       lea    -0x1e8(%rbp),%rsi
  400760:       mov    $0x0,%eax
  400765:       mov    $0x18,%edx
20 40076a:       mov    %rsi,%rdi
  40076d:       mov    %rdx,%rcx
  400770:       rep stos %rax,%es:(%rdi) "Buffer with ecx=24  memset(buff,24, \0)
  400773:       mov    $0x4008b4,%edi $0x4008b4="USER"
  400778:       callq  4005b0 <getenv@plt>
25 40077d:       mov    %rax,-0x200(%rbp)
  400784:       mov    -0x200(%rbp),%rdx
  40078b:       lea    -0x1f0(%rbp),%rax
  400792:       mov    %rdx,%rsi
  400795:       mov    %rax,%rdi
30 400798:       callq  400630 <strcat@plt>
  40079d:       lea    -0x120(%rbp),%rax = "/tmp/m3cool/"
  4007a4:       mov    %rax,%rsi
  4007a7:       mov    $0x4008b9,%edi $0x4008b9="%s"
```

```
4007ac:       mov    $0x0,%eax
4007b1:       callq  400620 <__isoc99_scanf@plt>
4007b6:       lea    -0x1f0(%rbp),%rax
4007bd:       mov    $0x4008bc,%esi  "w"
4007c2:       mov    %rax,%rdi
4007c5:       callq  400610 <fopen@plt>
4007ca:       mov    %rax,-0x1f8(%rbp)
4007d1:       cmpq   $0x0,-0x1f8(%rbp)
4007d9:       je     40080a <main+0xdd>
4007db:       mov    -0x1f8(%rbp),%rdx
4007e2:       lea    -0x120(%rbp),%rax3
4007e9:       mov    %rdx,%rsi
4007ec:       mov    %rax,%rdi
4007ef:       callq  4005e0 <fputs@plt>
4007f4:       mov    -0x1f8(%rbp),%rax
4007fb:       mov    %rax,%rdi
4007fe:       callq  4005c0 <fclose@plt>
400803:       mov    $0x0,%eax
400808:       jmp    40080f <main+0xe2>
40080a:       mov    $0x1,%eax
40080f:       mov    -0x18(%rbp),%rbx
400813:       xor    %fs:0x28,%rbx
40081c:       je     400823 <main+0xf6>
40081e:       callq  4005d0 <__stack_chk_fail@plt>
400823:       add    $0x1f8,%rsp
40082a:       pop    %rbx
40082b:       pop    %rbp
40082c:       retq
40082d:       nopl   (%rax)
```

Listing 1: Escreve em /tmp/USER

2 DESAFIO - Parte 2

O ambiente deve ser todo configurado através de gerenciador de configuração, o que deverá ser entregue é um repositório git contendo os arquivos de configuração que serão aplicados em uma máquina virtual "zerada". Caso necessário descrever como executar o processo de aplicação da configuração na máquina virtual. Ao final da tarefa e execução do processo, deveremos ter um ambiente funcional;

- É recomendado que o repositório git seja entregue com commits parciais, mostrando a evolução de seu código e pensamento. Caso prefira nos informe um url de git público ou então compacte todos os arquivos em um .tar.gz mantendo a pasta .git em conjunto;

- No ambiente deverá estar rodando uma aplicação node.js de exemplo, conforme código abaixo. A versão do node.js deverá ser a última versão LTS disponível em: https://nodejs.org/en/download/. A aplicação node abaixo possui a dependência da biblioteca express. Garanta que seu processo de bootstrap instale essa dependência (última versão estável disponível em: http://expressjs.com/) e rode o processo node em background. De uma forma dinâmica garanta que seja criado uma instância node para cada processador existente na máquina (a máquina poderá ter de 1 a 32 processadores);

Construa dentro de sua automação um processo de deploy e rollback seguro e rápido da aplicação node. O deploy e rollback deverá garantir a instalação das dependências node novas (caso sejam adicionadas ou alteradas a versão de algum dependência por exemplo), deverá salvar a versão antiga para possível rollback e reiniciar todos processos node sem afetar a disponibilidade global da aplicação na máquina;

- A aplicação Node deverá ser acessado através de um Servidor Web configurado como Proxy Reverso e que deverá intermediar as conexões HTTP e HTTPS com os clientes e processos node. Um algoritmo de balanceamento deve ser configurado para distribuir entre os N processos node a carga recebida;

- A fim de garantir a disponibilidade do serviço, deverá estar funcional uma monitoração do processo Node e Web para caso de falha, o mesmo deve reiniciar ou subir novamente os serviços em caso de anomalia;

- Desenvolva um pequeno script que rode um teste de carga e demostre qual o Throughput máximo que seu servidor consegue atingir;

- Desenvolva um script que parseie o log de acesso do servidor Web e deverá rodar diariamente e enviar por e-mail um simples relatório, com a frequência das requisições e o respectivo código de resposta (ex:5 /index.html 200);

- Por fim; rode o seu parser de log para os logs gerados pelo teste de carga, garantindo que seu script terá performance mesmo em casos de logs com milhares de acessos;