Manual de uso de ferramentas - projeto CIRD

Agostinho Brito

Sumário

Prefácio

Neste documento são apresentadas instruções de uso de ferramentas de reconhecimento de sistemas operacionais desenvolvidas no âmbito do projeto CIRD.

As ferramentas foram desenvolvidas em um sistema operacional do tipo GNU/Linux. A adaptação para outros tipos de sistemas operacionais ainda é objeto de estudo futuro. Para que as ferramentas funcionem, algumas bibliotecas e programas auxiliares com licenças de código livre precisam ser instalados para funcionar.

Chapter 1. Instalação de dependências

Neste projeto foram usadas as bibliotecas de programação `libpcap` e `libfann` para captura de dados e treinamento de redes neurais artificiais e a ferramenta `paping` (disponível em https://github.com/rampageX/paping) para gerar tráfego nos dispositivos sob teste.

O desenvolvimento das ferramentas foi realizado num sistema GNU/Linux Ubuntu 17.10 e ambas as bibliotecas já se encontram no repositório da canonical. Caso as ferramentas não estejam disponíveis na distribuição utilizada, elas precisam ser baixadas e instaladas manualmente.

No Ubuntu, as bibliotecas e seus arquivos de desenvolvimento podem ser instaladas através do seguinte comando:

```
$ sudo apt install libpcap-dev libfann-dev
```

A instalação da ferramenta `paping` por sua vez não se dá de forma semelhante. Paping é uma ferramanta de código livre criada por Mike Lovell capaz de gerar tráfego TCP em uma porta especificada pelo usuário. Embora a ferramenta tenha sido abandonada em 2013, a última versão atende com sucesso aos requisitos do projeto.

Na versão final da ferramenta completa, a geração de tráfego deverá ser feita por um módulo desenvolvido em python específico para esse fim.

A instalação do `paping` pôde ser feita com a seguinte sequência de comandos:

```
$ git clone https://github.com/rampageX/paping
$ cd paping
$ mkdir bin
$ make
```

O processo gera no subdiretório `bin` o arquivo executável `paping`, que é a ferramenta de geração de tráfego.

O paping inicia conexões TCP em intervalos regulares de 1s com o dispositivo sob teste. Para os experimentos realizados, notou-se que este último responde com dois pacotes TCP para cada um das combinações de pacotes enviados pelo `paping`: uma para aceitar a conexão e outra para notificar a finalização desta.

De acordo com a documentão do paping, seu uso se dá repassando o IP da máquina e algumas opções, como sugere a ferramenta:

```
paping v1.5.5 - Copyright (c) 2017 Mike Lovell

Syntax: paping [options] destination

Options:
 -?, --help display usage
 -p, --port N   set TCP port N (required)
     --nocolor  Disable color output
 -t, --timeout  timeout in milliseconds (default 1000)
 -c, --count N  set number of checks to N
```

O uso do paping foi feito conforme mostra o exemplo seguinte:

```
$ ./paping -c 100 -p 80 10.13.111.173
```

Neste exemplo, o paping envia **100** unidades de teste para a máquina com endereço IP **10.13.111.173** na porta **80** em intervalos de tempos igualmente espaçados de **um segundo**. Procurou-se utilizar opção de regular o *timeout* da ferramenta (que é fixada em 1s), mas não se logrou sucesso nessa etapa.

Chapter 2. Ferramentas de reconhecimento de camadas de rede e transporte

2.1. Preparação das ferramentas

Uma vez instaladas as bibliotecas e a compilada a ferramenta auxiliar, basta executar a seguinte sequência de comandos no diretório raiz

```
$ mkdir bin
$ make
```

Todas as ferramentas serão compiladas e os seus respectivos executáveis serão armazenados no diretório `bin`.

2.2. Apresentação da ferramenta de captura

a fazer... provavelmente joao paulo já tem isso disponível.

2.3. Apresentação das ferramentas de preparação do espaco de fase

As ferramentas de geração de tráfego geram como respostas dois tipos de informações: pacotes de respostas de sincronismo TCP (para análise dos números iniciais de sequência) e pacotes de resposta IP (para análise do campo IP ID). Em ambos os casos, uma sequência de números é gerada e analisada em busca da criação de uma assinatura para o sistema operacional que a gerou.

Nessa etapa foi necessário desenvolver duas ferramentas: uma para preparar o espaço de fase conforme as amostras coletadas do TCP ISN e outra para preparar o espaço de fase conforme as amostras de IP ID.

Cada ferramenta existe pela necessidade de atender a protocolos diferentes em situações de capturas distintas e segundo pela faixas de valores que cada campo prevê. O TCP ISN usa um `unsigned int` (32 bits) para armazenar informações, ao passo que o IP ID usa um `unsigned short`.

O estudo do espaço de fase prevê a análise dos incrementos que o gerador de números aleatórios do sistema operacional fornece ao longo do tempo (em intervalos regulares de resposta). Quando o incremento (sempre positivo) ocorre próximo ao limite superior da faixa prevista para a variável, o próximo número segue para o início dessa faixa, de sorte que a diferença entre o último e o primeiro número é negativa. Esse efeito é então compensado conforme o tipo de dado associado ao campo (`unsigned int` ou `unsigned short`).

Os dois protótipos ferramentas são denominados `espaco_fase_ipid` e `espaco_fase_tcpisn` e podem ser utilizadas da seguinte forma:

```
$ bin/espaco_fase_tcpisn [samples]
$ bin/espaco_fase_ipid [samples]
```

onde `samples` é um arquivo escrito em texto simples contendo uma sequência de inteiros coletados em uma sessão de captura, separados por espaços, quebras de linha ou tabulações.

Como produto, a ferramenta criará um arquivo denominado `samples.fas`, onde armazenará o espaço de fase. Nesse arquivo, cada linha conterá dois inteiros, contendo duas diferenças consecutivas entre números de sequências capturadas. O espaço de fase consiste portanto em uma representação bidimensional que relaciona um incremento anterior dado a um número de sequência com o incremento seguinte.

Considere o exemplo seguinte contendo um conjunto de quatro números iniciais de sequência.

Listagem 1. amostras_tcpisn.txt

```
100
113
135
155
```

O cálculo do espaço de fase para TCP ISN é obtido em se executando o comando

```
$ bin/espaco_fase_tcpisn amostras_tcpisn.txt
```

e produzindo o arquivo `amostras_tcpisn.txt.fas` com o espaço de fase contendo as sequências de diferenças entre os ISNs. Um procedimento bem parecido ocorre para a ferramenta `espaco_fase_tcpisn`, sendo idênticos os formatos de representação dos arquivos de entrada e saída.

Listagem 2. amostras_tcpisn.txt.fas

```
13 22
22 20
```

2.4. Apresentação da ferramenta para cálculo de descritores

A identificação dos sistemas operacionais pela análise do espaço de fase é feita com um método que esquadrinha o espaço de fase em intervalos regulares e prepara uma matriz com base nas ocorrências de pares de pontos dentro de cada região.

Tal matriz é analisada usando técnicas de análise de textura utilizadas em processamento digital de imagens que produzem para cada matriz um conjunto de seis medidas estatísticas:

1. Probabilidade máxima

2. Correlação entre células vizinhas
3. Contraste entre uma célula e sua célula vizinha
4. Uniformidade da matriz
5. Homogeneidade (que mede a proximidade da diagonal principal)
6. Entropia (que mede a aleatoriedade da matriz)

A ferramenta desenvolvida opera processando os espaços de fase obtidos com as ferramentas previamente descritas. Cada execução da ferramenta cria dois outros arquivos: um contendo os descritores associados com o espaço de fase e uma matriz de dimensões `32 x 32` elementos contendo a versão quantizada do espaço de fase, normalizada na faixa stem:$[0,1]$.

O primeiro arquivo contendo os descritores é denotado com a extensão `.desc`, cujo nome é criado automaticamente conforme o nome do arquivo passado como parâmetro para a ferramenta. O segundo arquivo contendo a matriz quantizada é denotado com a extensão `.quant`.

A execução da ferramenta para cálculo dos descritores é realizada com os seguintes comandos:

```
$ bin/descriptors [input.fas]
```

onde `input.fas` é o arquivo contendo os pares de pontos do espaço de fase. Por exemplo, a execução do comando `bin/descriptors windows10.fas` geraria os arquivos `windows10.fas.desc` contendo os descritores e o arquivo `windows10.fas.quant` com a matriz quantizada.

Os arquivos contendo matrizes quantizadas geradas pela ferramenta na presente etapa do projeto servem apenas como insumo para análise e criação de documentação científica. Não possuem utilidade no processo de classificação e identificação dos sistemas. Em tempo, as matrizes são guardadas em formato de ponto flutuante, organizadas no arquivo em linhas consecutivas, cada coluna separada da outra por um espaço.

Os arquivos com os descritores, por sua vez contêm sete campos separados por espaços, sendo o primeiro campo um identificador com o nome do arquivo que guarda o espaço de fase e o restante os descritores calculados para a matriz quantizada, organizados na seguinte sequência: **Probabilidade máxima**, **Correlação**, **Contraste**, **Uniformidade**, **Homogeneidade** e **Entropia**. Por exemplo, para uma sequência de captura gerada para o sistema operacional Windows 8, o seguinte arquivo foi produzido pela ferramenta de cálculo de descritores:

```
in/windows-8-large.fas 0.00370879 0.00248084 53.3223 0.00299652 0.244457 5.863
```

2.5. Preparação e treinamento de Redes Neurais Artificiais para

reconhecimento de sistemas operacionais ===

A identificação dos Sistemas Operacionais pela análise das camadas de transporte e rede no âmbito do projeto CIRD foi feita através dos descritores obtidos pelas matrizes quantizadas conforme

descrito em seção anterior.

Cada conjunto de dados de um espaço de fase produz um arquivo contendo seis descritores associados com a matriz quantizada do espaço de fase.

Dada a multidimensionalidade do vetor de descritores, optou-se pelo uso de uma Rede Neural Artificial do tipo Perceptrons de Múltiplas Camadas para associar um descritor obtido de um dispositivo sob teste a um sistema operacional previamente identificado.

O processo treinamento das redes neurais (e posterior uso para reconhecimento) foi realizado usando a `libfann`, uma biblioteca de programação livre distribuída sob a licença GNU LESSER GENERAL PUBLIC LICENSE.

A preparação dos dados foi realizada com base em um conjunto de dados coletados de sistemas operacionais diversos, onde para cada sistema um conjunto de aproximadamente 65.000 amostras de TCP ISNs e 30.000 amostras de IP iDs foram coletadas. Tal quantidade de amostras foi selecionada para conseguir montar uma representação fiel de cada sistema e obter descritores de boa qualidade. O conjunto de amostras usados para treinamento foi rotulado no projeto com a terminação `large` para fins de identificação.

Nos processos de reconhecimento de sistemas em tempo real, amostras menores poderão ser usadas para fins de comparação, posto que o processo de reconhecimento poderá ser feito de forma incremental.

Primeiramente, todos os descritores usados para caracterizar cada sistema operacional devem ser calculados e em seguida aglomerados em um único arquivo para preparar o conjunto de treinamento.

O exemplo mostrado nas linhas seguintes mostra como treinar uma Rede Neural para identificar sistemas pelos espaços de fase gerados pelos campos TCP ISN.

Assumindo, por exemplo, que os arquivos dos descritores tenham sido gravados no subdiretório `training_tcpisn`, a preparação do conjunto de treinamento pode ser feito com a seguinte sequência de comandos:

```
#!/bin/sh
for file in training_tcpisn/*large; do
        bin/espaco_fase_tcpisn $file;
done

for file in training_ipid/*large; do
        bin/espaco_fase_ipid $file;
done

for file in training_tcpisn/*large.fas; do
        bin/descriptors $file;
done

for file in training_ipid/*large.fas; do
        bin/descriptors $file;
done

cat training_tcpisn/*large.fas.desc > so-tcpisn-large.txt
cat training_ipid/*large.fas.desc > so-ipid-large.txt

bin/prepare_data so-tcpisn-large.txt so-tcpisn-large-input.txt labels_tcpisn.txt
bin/prepare_data so-ipid-large.txt so-ipid-large-input.txt labels_ipid.txt

bin/train_mlp so-tcpisn-large-input.txt so_neural_tcpisn.net
bin/train_mlp so-ipid-large-input.txt so_neural_ipid.net
```

No exemplo, os dois primeiros loops do script varrem todos os arquivos com as amostras presentes nos subdiretórios `training_tcpisn` e `training_ipid` e calculam os espaços de fase para cada um.

Os dois loops seguintes calculam os descritores de cada um dos espaços de fase presentes nos arquivos com extensão `.fas`

Em seguida, todos os descritores calculados na etapa anterior são aglomerados em dois arquivos: `so-tcpisn-large.txt` e `so-ipid-large.txt`.

Para o conjunto de dados utilizado no exemplo, o seguintes arquivos foram produzidos.

Listagem 3. so-tcpisn-large.txt

```
in/3com-4210-large.fas 0.100009 0.0861942 48.3753 0.0145694 0.309485 5.19855
in/freebsd-9.2-large.fas 0.027955 -2.32568 57.9164 0.00584189 0.245131 5.42946
in/hp-printer-large.fas 1 0 0 1 1 -0
in/ios-12.3.11-large.fas 0.00385407 -0.0191657 53.7561 0.00294808 0.241621 5.85158
in/linux-2.6.32-large.fas 0.693492 -2.11676 13.8228 0.490513 0.821034 1.67564
in/linux-3.2.29-large.fas 0.276931 0.575046 1.16984 0.185536 0.757536 2.16554
in/mac-os-x-10.8.4-large.fas 0.0149083 -4.51369 62.5106 0.0083262 0.232563 4.92356
in/netbsd-5.1.2-large.fas 0.00656297 -11.5486 74.1994 0.00331893 0.210819 5.90248
in/openbsd-4.3-large.fas 0.00387585 -0.0213783 53.7451 0.00298447 0.243014 5.84269
in/plan9-4-large.fas 0.0137638 0.0391498 51.6788 0.00318794 0.25481 5.80708
in/qnx-6.5-large.fas 0.00647314 -11.2204 73.3783 0.00332539 0.211988 5.89817
in/solaris-11.1-large.fas 0.00642415 -13.4096 80.3753 0.00312933 0.205413 5.90957
in/sonicwall-2004-large.fas 0.385464 -5.94786 65.4814 0.158117 0.478118 3.06906
in/windows-8-large.fas 0.00370879 0.00248084 53.3223 0.00299652 0.244457 5.863
in/windows-xp-large.fas 0.010301 -1.50719 48.7784 0.0072789 0.305373 5.13473
in/xerox-printer-large.fas 1 0 0 1 1 -0
```

Listagem 4. so-ipid-large.txt

```
training_ipid/android-s7-large.fas 0.122661 -2.80127 57.9906 0.0255704 0.294972
4.41478
training_ipid/ubuntu-17.10-large.fas 0.0311134 -13.2464 80.1595 0.0153414 0.210234
4.99282
training_ipid/windows10-large.fas 0.93041 -0.092847 6.10068 0.866833 0.937877 0.363894
training_ipid/windows8.1-large.fas 1 0 0 1 1 -0
training_ipid/windowsXP-large.fas 0.98183 -0.00684642 1.74451 0.964072 0.983839
0.116335
```

A ferramenta `prepare_data` segue com a preparação dos dados para treinamento. Ela opera com três argumentos: o primeiro é o conjunto de descritores algomerados obtido na etapa anterior; o segundo, produzido como saída pela ferramenta é o nome do arquivo que será usado para treinar a rede neural; e o terceiro é um arquivo de texto contendo os rótulos dos sistemas operacionais usados no treinamento. Esse último arquivo será necessário na pela ferramenta de reconhecimento de sistemas.

Finalmente, a última linha do script invoca o processo de treinamento da rede neural. O primeiro argumento fornecido à ferramenta deve conter o arquivo preparado para treinamento (no caso, os arquivos `so-tcpisn-large-input.txt` e `so-ipid-large-input.txt`) e produz na saída a estrutura da rede neural (denotado no exemplo por `so_neural_tcpisn.net` e `so_neural_ipid.net`).

O reconhecimento do sistema operacional de um dispositivo sob testes pode ser feito com a ferramenta `find_so` desenvolvida para esse fim.

```
$ bin/find_so so_neural_tcpisn.net labels_tcpisn.txt training/linux-2.6.32-
large.fas.desc
$ bin/find_so so_neural_ipid.net labels_ipid.txt training/linux-2.6.32-large.fas.desc
```

O primeiro argumento fornecido é o arquivo contendo a estrutura da rede neural treinada (`so_neural_tcpisn.net` ou `so_neural_ipid.net`); O segundo é a lista de rótulos gerada na etapa de preparação de dados (`labels_tcpisn.txt` ou `labels_ipid.txt`); e o último argumento é um arquivo contendo os descritores do dispositivo a ser identificado. Neste caso, deve-se coletar previamente as amostras desse dispositivo, calcular o espaço de fase e respectivos descritores para proceder com o teste de identificação.