TÉCNICO LISBOA

Mestrado em Engenharia Informática e de Computadores
1º Semestre 2013/2014

Relatório de projecto - Segurança Informática em redes e sistemas

Detecção de Software Malicioso baseado em comportamentos

Grupo 12:
- André Rodrigues nº 69879
- Carlos Bartolomeu nº 69656
- Margarida Silva nº 66936

Tópicos a abordar:

1. Arquitectura e suas limitações;
2. Obtenção dos recursos do sistema operativo (utilização do SIGAR) tendo em conta os tipos de vírus que queremos detectar;
3. Classificação supervisionada;
4. Classificador escolhido;
5. Criação dos vírus;
6. Conclusão;

1. Arquitectura:

.

Criámos dois módulos, um Scanner e um *Analyser*.

O *Scanner* utiliza a ***SIGAR API*** para obter informação sobre os processos a correr, recolhendo dados sobres os recursos do sistema operativo a serem utilizados por estes num dado momento. Pode ser corrido de dois modos: no modo de aprendizagem gera um ficheiro com os dados de treino, e no modo de análise envia os dados ao Analyser via Socket.
Utilizámos também alguns comandos *Linux*, como por exemplo o 'lsof' para obter os processos que estão a utilizar determinado descritor de ficheiro.

O Analyser comunica com o Scanner através de uma ligação ***TCP ,*** analisando os dados enviados pelo Scanner e por sua vez aplica o algoritmo de classificação a estes, produzindo um resultado que indica se um dado processo é virus ou não.

Uma das limitações do nosso programa é este só correr em ambiente *Linux*.

Nota: Optámos por implementar comunicação via socket entre os componentes *Scanner* e *Analyser* por uma questão de modularização.

2. Obtenção dos recursos do sistema operativo (utilização do SIGAR) tendo em conta os tipos de vírus que queremos detectar:

Como queremos detectar vírus que influenciem o **CPU**, a Internet e a Webcam, tivemos de recolher o maior número possível de variáveis que pudessem ser influenciados na presença destes vírus. Por isso, alguns dos dados recolhidos são:

- ➔ Memória utilizada;
- ➔ Percentagem de CPU utilizada;
- ➔ Estado do processo (Running, Sleeping, Dead);
- ➔ Uso de Internet;
- ➔ Uso da Webcam;
- ➔ Tempo desde o inicio do processo;
- ➔ Se o processo é *root*;
- ➔ Se o programa é conhecido.

Por programa conhecido entenda-se todos os programas que achámos serem utilizados com mais

frequência. A escrita dos nomes dos programas foi feita manualmente num ficheiro, que posteriormente será consultado exactamente para identificar um programa como sendo conhecido ou não, e foi feita com base nos programas que estavam a ser executados aquando da fase de treino.

3. Classificação supervisionada:

Quando queremos treinar um algoritmo de aprendizagem é necessário recolher uma amostra considerável de dados , no caso particular do nosso projecto , este dados são previamente obtidos pelo Scanner que os coloca num ficheiro (ficheiro de treino). Para concretizar o treino , recorremos a uma ferramenta (WEKA) que utiliza um classificador (algoritmo de aprendizagem supervisionada). Uma vez que utilizamos aprendizagem supervisionada, para uma amostra de dados relativa a um processo, é necessário indicar se este se trata de um vírus ou não. Por fim, o classificador treinado, também chamado de modelo, é guardado num ficheiro que posteriormente será utilizado no modo avaliação.

4. Classificador escolhido:

O algoritmo escolhido para aprendizagem supervisionada foi o *Multilayer Perceptron* (redes neuronais). A escolha recaiu sobre este porque: é possível olhar para o modelo gerado e perceber que variáveis estão a ser utilizadas e como influenciam a previsão, e de todos os algoritmos experimentados foi o que produziu melhores resultados. Entre os algoritmos testados encontram-se *árvores de decisão*, *naive bayes*, *Logistic Regression*.

O resultado produzido por um dado classificador deve estar num conjunto bem definido e limitado, e no caso particular do nosso projecto encontra-se no conjunto de valores {true, false}, true se um dado processo é vírus e falso caso contrário.

5. Criação dos vírus:

De forma a testar a eficácia do nosso programa, foram feitos 3 vírus em que cada um se foca na utilização excessiva de diferentes recursos. Os vírus são:

- **CPUVirus** - consiste num ciclo infinito;
- **WebcamVirus** - guarda de 150 em 150 frames uma fotografia utilizando a webcam e recorrendo à utilização do *VLC* (VideoLan Codec);
- **NetworkVirus** - foi criado um esquema cliente-servidor em que o cliente envia de 50 em 50 milisegundos informação aleatória lida do /dev/urandom.

6. Conclusão

Executámos bastantes programas para testar, de entre os quais são:

- Mozzilla Firefox a correr vídeos no Youtube;
- Gnumeric a consumir 100% de CPU;

- ➔ Programas a utilizarem a webcam;
- ➔ Mozzilla Firefox a fazer download de ficheiros grandes (4Gb) a alta velocidade;
- ➔ Os jogos FreeCell e Mastermind;

O programa funciona como previsto, ou seja, detecta apenas os vírus criados quando estes estão a correr. Não existem nem falsos-positivos nem falsos-negativos, o que não implica que não possam aparecer, tendo em conta que não testamos exaustivamente.

Durante o desenvolvimento do programa, tivemos alguns problemas na obtenção de recursos do sistema operativo porque apesar de termos encontrado a SIGAR API, a documentação da mesma é inexistente.

Em relação à ferramenta WEKA, existe bastante documentação pelo que facilitou bastante a sua utilização.

De forma a correr o projecto, existe um ficheiro README na directoria do projecto que explica todos os passos necessários para a execução do mesmo.