master 1 Branch 0 Tags Go to file Go to file Code

aparisot84 Comlementação do Readme para contemplar os avanços da pesquisa para d... 10 months ago

.idea	Comlementação do Readme para contempl...	10 months ago
1 - Scripts	Status das seções após a transformação em...	2 years ago
2 - HashList	Primeiro commit do projeto.	2 years ago
License.md	Create License.md	2 years ago
Readme.md	Comlementação do Readme para contempl...	10 months ago
Requirements.txt	Primeiro commit do projeto.	2 years ago

README GPL-3.0 license

Montagem de Dataset para Detecção de Ataques de Ransomware: Desafios e Soluções

STATUS EM DESENVOLVIMENTO LANGUAGE PYTHON3

DISCLAIMER: Os scripts contidos neste projeto fazem o download de malware direto para a sua máquina. A sintaxe das pastas (com //) é a do linux (onde nativamente os sripts poderiam ser executados) e os arquivos baixados são apenas DLL ou EXE. Se você não tem plena ciência do que está fazendo, não execute os scripts, pois corre o risco de ter seus arquivos criptografados e não me responsabilizo se isso acontecer.

1 - Decrição do Projeto:

Este projeto surgiu a partir do meu trabalho de dissertação de mestrado na área de Segurança de Sistemas na Universidade Federal Fluminense. Durante o desenvolvimento da pesquisa, tive a necessidade de procurar amostras de ransomware nos repositórios diponíveis na internet, como VirusShare, VirusTtal e malware Bazaar e construir scripts que pudessem automatizar este processo.
Dentro do escopo geral, cada script realiza uma tarefa específica. Preferi manter dessa maneira, pois achei importante conseguir fazer verificações intermediárias no processo.

2 - Requisitos e limitações:

Os repositórios utilizados nos scripts são o VirusTotal (VT), MalwareBazzaar (MB) e VirusShare (VS). Para realizar consultas e downloads de amostras, você deve se cadastrar nesses repósitórios e gerar sua chave API. Esta chave que vai te permitir os scripts interagirem com os repositórios.
Uma observação que cabe ser feita é que o VirusTotal não permite download de amostras de usuários comuns (mesmo com assinatura acadêmica), apenas usuários vinculados a empresas e que tenham assinatura. Por este motivo precisei utilizar outros repositórios para fazer o download das amostras. Atente-se também para as limitações impostas pelos repositórios para interações com suas API (se não houvesse limites, poderiamos causar um DoS):

```
VirusTotal (Licença Educacional/Pesquisa):
    Request rate => 1000 lookups/min
    Daily quota => 20 K lookups/day
    Monthly quota => 620K lookups/month

VirusShare:
    Request rate: 4 requests/min
    Daily Quota: 5,760 requests
    Monthly Quota: 172,800 requests
```

3 - Instalação e uso:

Basta clonar o repositório para sua máquina e escolher o malware que deseja procurar (alterando o nome no arquivo 1 - Ransomware HashList Download.py). Recomenda-se manter a estrutura das pastas. Para conseguir reproduzir meus passos até o final, você necessitará ter o cuckoo sandbox funcionando. O arquivo 'Guia Cuckoo.txt' é um passo a passo para instalação e execução do cuckoo que consegui aqui e adicionei alguns passos extras que necessitei ao realizar este trabalho.

4 - Scripts:

4.1 - Ransomware HashList Download:

A partir do nome de um malware que se deseja conseguir amostras, este script baixa os hashes encontrados no VT e os grava em arquivos distintos para cada malware na pasta 'HashList'.

4.2 - HashList Submit Downoad Sample

Este script verifica o conteúdo da pasta HashList e checa os dados no VT para arquivos DLL ou EXE e que tenham o nome do malware na chave 'suggested_threat_label'. Caso atenda os requisitos de formato e nome, o script procura amostras disponíveis no VS e MB e grava na pasta 'ZIP Samples & Download Logs', juntamente com a situação de cada hash: se foi descartado pela extensão ser diferente e/ou se foi e encontrado (ou não) nos repositórios.

4.3 - Unzip Samples Download Reports

Arquivo que deszipa as amostras de arquivos baixados dos repositórios utilizados. O arquivo tabmém tem a função de baixar os relatórios em JSON do cuckoo. Tenha em mente que os relatórios podem ter de alguns MB até vários GB, dependendo das interações que o malware faz com a Máquina Virtual usada para análise.

4.4 - Dataset Constructor from Json Reports

Este script transforma os arquivos dos relatórios, em um dataframe pandas, convertendo os dados selecionados em features para o dataset a partir da abordagem da quantidade de chamadas de API.

4.5 - ML Explorer

(acho que isso foi um rascunho q fiz pra testar uns algiritmos)
Este é um arquivo Jupyter Notebook que faz tratamento do dataframe, como transformar o que é NaN em zeros (zero chamada àquela API). Ao final, o dataset gerado estará pronto para ser utilizado em classificadores de ML.

4.6 - JSON to TXT

Script que carrega os arquivos JSON e transforma em arquivos de texto as entradas da seção selecionada.

4.7 - TXT to Dataframe

4.8 - JSON to TXT(split)

4.9 - Section Strings to Dataframe

Transforma os arquivos

4.10 - API Count Ransomware Detection

4.11 - TF-IDF Ransomware Detection

Cabe observar que os arquivos "JSON to TXT" e XXXX, que fazem a filtragem dos dados dos relatórios e grava em formato diferente estão com os trechos de seleção dos dados comentados de acordo com a sua utilização. O exemplo abaixo, retirado do arquivo "JSON to TXT", mostra como está organizado. Nele podemos ver que os trechos de código que extraem as informações das seções extracted e signatures estão comentadas, ao passo que a extração de dados da seção behavios está ativa.

```
"""
if ("extracted" in json_file):
    for i in range(len(json_file["extracted"])):
        for key in ["category", 'raw', 'program']:
            filtered_json_file["extracted" + str(i)] = json_file["extracted"][i][key]
...
if ("signatures" in json_file):
    for i in range(len(json_file['signatures'])):
        filtered_json_file['signatures'] = json_file['signatures'][i]['description']
"""
```

```
    if ("behavior" in json_file): filtered_json_file["behavior_summary"] = json_file["behavior"]["summary"] if (
                "summary" in json_file["behavior"]) else None
```

5 - Datasets

Após o processamento dos relatórios gerados pelo cuckoo, tanto para os ransomwares quanto para os softwares benignos, foram gerados os conjuntos de dados descritos abaixo, divididos em dois grupos:

1. Grupo com o conjunto de dados formados a partir da abordagem de chamadas de API:

Este grupo é formado pelo conjunto de dados 1075.csv. Neste arquivo cada linha representa uma amostra e as colunas representam as chamadas de API executadas na interação com o SO. O dataframe é populado com as quantidades de chamadas que determinado ransomware fez durante sua atividade.

2. Grupo com conjuntos de dados populados com dados resultantes da transformação dos relatórios em texto e aplicação do TF-IDF:

Conforme será explicado com detalhes na próxima seção, devido a capacidade computacional disponível, não foi possível trabalhar com um conjunto de dados único a partir da aplicação da abordagem TF-IDF (arquivo Behavior.csv, com 3,95GB). Por esse motivo, houve a necessidade de particionamento dos dados e a criação de conjuntos de dados reduzidos. A execução dessa tarefa resultou em conjuntos de dados menores, representativos das seções Behavior, Network, Memory, Signatures e Strings.

Adicionalmente, como a seção behavior é a seção dos relatórios que contém a maior parte da interação dos ransomwares com a VM, mesmo com a separação dos dados em seções, seu tamanho continuou aquém da capacidade de memória disponível e, desse modo, optei por separar também por família de malware. Cabe ressaltar que as amostras benignas analisadas estão contidas em cada arquivo.

O resuldado das divisões são os arquivos listados abaixo:

Behaviorccel.csv -> seção behavior dos relatórios das amostras do Conti, Egregor, Clop e LockBit.

Behaviormnr.csv -> seção behavior dos relatórios das amostras do MountLocker, Netwalker e Ryuk.

Behaviorrevil.csv -> seção behavior dos relatórios das amostras do Revil.

Caso deseje utilizar os conjuntos de dados gerados por este trabalho, fique à vontade. A pasta com os conjuntos de dados pode ser acessada aqui.

5 - Classificação

Cabe ressaltar que as amostras benignas analisadas estão contidas em cada arquivo.Para realizar a classificação foram aplicados algoritmos de Aprendizado de Máquina conjuntamente com redutores de dimensionalidade (PCA) e padronizadores (Standard Scaler).

Para descobrirmos qual a melhor configuração para classificação dentro de intervalos pré definidos, utilizamos o Grid Search CV.

6 - Limitações Técnicas

A principal limitação deste trabalho foi conseguir processar os relatórios para transformá-los em conjuntos de dados e depois carregar esses conjuntos de dados para utilização nos algoritmos de Aprendizado de Máquina. A máquina disponivel para os testes era um Intel® Core™ i7-8550U CPU @ 1.80GHz × 8 com 16 GB RAM DDR4 rodando um Ubuntu 20.04.4 LTS (64 bits). Apesar de ser uma máquina doméstica relativamente robusta e com uma boa quantidade de memória, houve a necessidade de aumentar a memória virtual do S.O. (coloquei 50GB, um exagero, mas eu queria evitar ter que ficar testando vários valores) para poder fazer a transformação dos dados brutos em conjuntos de dados. Apesar de a solução ter resolvido para a transformação dos dados, quando tive que efetivamente carregar os datasets em memória para fazer as classificações, essa solução não foi suficinte e a saída que encontrei foi assinar o Google Colab Pro+, que libera 52GB de RAM para a máquina. Com essa quantidade de RAM foi possível carregar os dados e fazer a classificação, mas mesmo assim tive que dividir os conjuntos de dados pelas seções e a seção behavior teve também que ser dividida por família de ransomware.

OBS: Os arquivos TF-IDF e MLExplorer foram utilizados como rascunhos de implementação durante o processo de elaboração dos scripts.