Engenharia de Segurança

Vulnerabilities Mapping

Raphael Jubram Sawaia Pinheiro (pg37160@alunos.uminho.pt)

Gestão de Vulnerabilidades

Uma das principais atividades de equipes de Segurança da Informação é fazer a Gestão de Vulnerabilidades. Nesse trabalho, os analistas de segurança precisam identificar vulnerabilidades nos sistemas e tecnologias utilizados pela empresa e definir um plano de correção, com o objetivo de minimizar os riscos de ataques cibernéticos.

As vulnerabilidades podem ser identificadas de diversas formas, das quais, as mais comuns são:

- Varreduras de Vulnerabilidades;
- Testes de Intrusão (penetration tests);
- Programas de Bug Bounty;
- Contribuição de usuários;
- Pesquisas internas.

Ferramentas de Varredura

As varreduras de vulnerabilidades fazem parte do processo contínuo de gestão de vulnerabilidades para empresas preocupadas com sua segurança cibernética. Existem diversas ferramentas especializadas em realizar esse tipo de tarefa, das quais podemos citar algumas das mais famosas:

- Nessus
- Qualys
- Acunetix
- OpenVas
- Nexpose

Cada uma dessas ferramentas possui seu próprio motor de varredura, e identifica as vulnerabilidades de uma maneira diferente. Os fabricantes possuem bancos de dados de vulnerabilidades, nos quais armazena diversas informações sobre cada ameaça que sua ferramenta verifica. A representação de uma vulnerabilidade por uma ferramenta é denominada *plugin*. No escopo do presente trabalho, iremos utilizar os *plugins* do Nessus e do Qualys.

Nessus

O Nessus é uma das principais ferramentas de varredura de vulnerabilidades, e é desenvolvido pela Tenable [1]. Atualmente, o Nessus conta com quase 120 mil *plugins*, que cobrem quase 50 mil CVE IDs [2]. Um *plugin* do Nessus é composto, dentre outros, pelos seguintes atributos:

```
Id: identificador único do plugin
Name: título do plugin
BugTraq Id(s): identificador único do BugTraq
CVE Id(s): lista de CVE IDs associados ao plugin
Category: classificação do tipo da vulnerabilidade
Family: categoria a que o plugin pertence
Synopsis: breve descrição
Description: longa descrição
Severity: severidade do plugin, que varia de Informativo a Crítico
Published: data de publicação do plugin
Modified: última vez que o plugin foi publicado
Version: versão atual do plugin
X-Reference(s): outras referências associadas ao plugin
```

Exemplo de *plugin* Nessus:

```
Id: 10669
Name: A1Stats Multiple Script Traversal Arbitrary File Access
BugTraq Id(s): 2705
CVE Id(s): CVE-2001-0561
Category: remote
Family: CGI Abuses
Severity: Medium
```

Qualys

O Qualys é outra ferramenta de varredura de vulnerabilidades, desenvolvido pela empresa homónima [3]. Sua base é composta por cerca de 34 mil *plugins*, que possuem os seguintes atributos:

```
QualysID: identificador único do plugin
Title: título do plugin
Sub Category: lista de categorias mais genéricas do plugin
Category: categoria mais específica do plugin
CVE ID: lista de CVE IDs associados ao plugin
Vendor Reference: identificador da vulnerabilidade nos boletins do fabricante da
ferramenta vulnerável
CVSS: severidade do plugin
Bugtraq ID: identificador do BugTraq
Published: data de publicação do plugin
Modified: última vez que o plugin foi publicado
```

Exemplo de *plugin* do Qualys:

```
QualysID: 10340
Title: Drummon Miles A1Stats Directory Traversal Vulnerability
Sub Category: Remote Discovery, Patch Available, Exploit Available
Category: CGI
CVE ID: CVE-2001-0561
CVSS: 7.5
Bugtraq ID: 2705
```

Vulnerabilidades Duplicadas

Como pode ser visto no exemplo acima, ambas as ferramentas possuem sua própria representação da mesma vulnerabilidade. Quando uma empresa utiliza mais de uma ferramenta de varredura de vulnerabilidades, corre o risco de elencar as vulnerabilidades em duplicidade. O problema é que a correlação entre os diferentes *plugins* não é trivial. Apesar de haver, em alguns casos, bastante similaridade entre os *plugins*, em outros eles possuem informações bastante diferentes.

Mapeamento de Plugins de Diferentes Fontes

Para ajudar a sanar esse problema, a solução desenvolvida utiliza o processamento da linguagem natural para classificar a similaridade entre *plugins*. A princípio, apenas três atributos são utilizados para a comparação: o título do plugin, a lista de CVE IDs e as Referências externas (incluindo o BugTraq ID).

A solução utiliza a linguagem de programação Python3, o banco de dados mongoDB e as bibliotecas Pandas [4], NLTK [5] e Fuzzywuzzy [6].

Instalação

Para utilizar a ferramenta, é necessário instalar as ferramentas mencionadas. A instalação das bibliotecas Python pode ser feita de forma automatizada, acessando o diretório da aplicação, com o comando:

```
pip3 install -r requirements.txt
```

Utilização

A ferramenta está dividida em 4 partes: `database.py`, `parser.py`, `plugin.py` e `utils.py`. Para utilizar a ferramenta, aconselha-se abrir o *prompt* Python no terminal e interagir com a ferramenta em tempo real. A interação é feita pelos comandos do módulo `Utils`, dessa forma, após acessar o *prompt* do Python, é necessário importar esse módulo:

```
from utils import Utils
```

Construção do Banco de Dados

Com o mongoDB instalado e a rodar na máquina local, e após o import do módulo `Utils`, basta rodar os comandos abaixo para construir o banco de dados:

```
# Irá construir uma base com cerca de 19 mil plugins
# verbose=True mostrará o progresso, que pode demorar um pouco
Utils.build_db(verbose=True)

# Irá construir uma base menor para testes
Utils.build_test_db()
```

Geração de Métricas

Com a base de dados concluída, é possível gerar as métricas baseadas na Matriz de Confusão, com o seguinte comando:

```
Utils.metrics()
```

Conclusão e Trabalho Futuro

Com a realização desse trabalho, foi possível perceber um grande potencial da ferramenta para solucionar o problema identificado. Na versão atual, a ferramenta ainda apresenta algumas falhas, principalmente no tocante a *True Negatives* e *False Negatives*, o que afeta a métrica do *Recall*. Porém, foi possível vislumbrar a medida de *Precision* de 90%, o que é um resultado bastante expressivo.

Como trabalho futuro, é pertinente analisar os *thresholds* para *matches* e *não matches*, bem como ajustar a fórmula matemática que faz o cálculo da similaridade. Além disso, é possível analisar outros atributos dos *plugins*, além dos três que foram utilizados nesse trabalho. Por fim, a ferramenta poderá ser melhorada consideravelmente utilizando técnicas de *Machine Learning* e *Data Mining*.

---

1. https://www.tenable.com/products/nessus/nessus-professional↩
2. https://www.tenable.com/plugins↩
3. https://www.qualys.com/apps/vulnerability-management/↩
4. https://pandas.pydata.org/↩
5. https://www.nltk.org/↩
6. https://github.com/seatgeek/fuzzywuzzy↩