

# UMA PLATAFORMA PARA MONITORIZAÇÃO DE SEGURANÇA EM AMBIENTES DOMÉSTICOS

Bruno Rodrigues, pg41066 – Carlos Alves pg41840 – Roberto Cachada a81012

Universidade do Minho  
pg41066@alunos.uminho.pt  
pg41840@alunos.uminho.pt  
a81012@alunos.uminho.pt

**Abstrato.** Este projeto tem por objetivo principal o desenvolvimento de uma plataforma modular para monitorização de propriedades de segurança da informação em ambientes domésticos. Como objetivo específico, será criado um ambiente *user friendly* que permita aos utilizadores catalogar a infraestrutura doméstica e receber alertas sobre possíveis violações de segurança ou sobre a publicação de novas vulnerabilidades de alta criticidade.

## 1 Introdução

Neste trabalho para Laboratório em Engenharia Informática, foi-nos pedido para desenvolver uma aplicação que pudesse encontrar vulnerabilidades nos programas instalados, num ambiente doméstico. Para tal, utilizamos a base de dados da NIST para obter informações sobre CVE, utilizando CPE obtidos a partir das informações dos programas instalados, de maneira a ser possível fazer *matching* com os CVE correspondentes.

Este trabalho consistiu na compilação de scripts escritos em *Python* que fazem o seguinte:

- Obtêm os programas instalados – Foi desenvolvido uma versão para Windows e Ubuntu para esta parte;
- Tratam as informações obtidas – Este processo consiste em retirar caracteres especiais, pôr todas as *strings* em minúsculas, entre outros;
- Construir um CPE – De maneira a ser possível fazer match com o CVE;
- Fazer vários *requests* à API da NIST – De modo a obter as informações necessárias (CVE, CVSS);
- Gravar numa base dados as informações obtidas;

E ainda, de modo a facilitar a execução deste programa foi desenvolvida uma Interface Gráfica (GUI).

Devido aos vários problemas relacionados com os CPE's, abordaremos esse fator mencionando as várias razões pelas quais obter um relatório de vulnerabilidades usando os CPE não é algo prático, e como tal não seria viável numa aplicação real.

## Estrutura

Uma das preocupações iniciais foi a forma de como se poderíamos obter as vulnerabilidades dos programas instalados, sem recorrer a outras ferramentas existentes. Após alguma pesquisa, descobrimos que é possível fazer *matching* de um CPE com um CVE.

Um CPE é um esquema de nomeação de software, que faz referência ao programa e a sua versão, que permite um *matching* direto com CVE associados. Na teoria, isto permitiria obter vulnerabilidades através da informação encontrada no sistema, algo que não é tão simples na prática, como vai ser mencionado posteriormente.

Primeiramente tentamos também obter todos os CVE existentes, e a sua informação (CVSS, CPE, entre outros) e colocá-las numa base de dados local. Isto provou ser demasiado lento, como também criava uma base de dados com um tamanho demasiado grande, e como tal, não foi utilizado. Depois de alguma leitura referente à API da NIST, descobrimos que era possível fazer pedidos ao website através da API fornecida, que nos permitia obter informações utilizando o CPE como *keyword* de referência. Como tal, tornou-se a solução ideal, que mesmo não sendo um processo rápido, é deveras mais rápido do que a maneira anterior, como também não necessitamos de ter toda a base de dados de CVE existentes no sistema local.

Em termos de armazenamento da informação, obtemos pela utilização do *sqlite*, uma base de dados local, sendo que esta é de fácil manejo, além que já está incluída com o *Python*, não sendo necessário instalar nada.

## Obter os programas

De maneira a obter os programas instalados, foram utilizados diferentes métodos nos dois sistemas operativos, Windows e Linux.

Para Linux foram utilizadas ferramentas pré-existentes no Linux como o *dpkg* ou através do *apt*.

Para Windows, foi observado que a maioria dos programas instalados armazenam varias informações sobre si mesma no Registro do Windows. Geralmente no seguinte local: `HKEY_LOCAL_MACHINE \ Software \ Microsoft \ Windows \ CurrentVersion \ Uninstall`

Porém alguns dos programas instalados não colocam informações no Registro, mas visto que grande parte o faz e para realizar uma busca por TODOS os programas seria necessário realizar buscas por incompatibilidades em relação ao que está no Registro ou não, além disso foi necessário utilizar módulos/extensões OS, neste caso o *Windows Management Instrumentation*(WMI). Mas visto que foram detetados alguns problemas ao encontrar tais arquivos, utilizamos *ctypes*.

Por fim, com estes métodos desenvolvidos conseguimos obter maior parte dos programas instalados em cada máquina, informações dos mesmos como: Nome do Produto, Código do Pacote, Versão, RegOwner, entre outros. E ainda para estender um

pouco mais este as funcionalidades deste trabalho, a obtenção das bibliotecas *Python* instaladas, para posteriormente serem verificadas os CVEs.

## Interface Gráfica GUI

Como já referido, para este projeto desenvolvemos duas interfaces gráficas. Primeiramente decidimos utilizar o *tkinter* sendo que esta é vista como sendo uma ferramenta simples e de fácil aprendizado incorporado em *Python*, portanto foi desenvolvido um visual simples com apenas um botão que permite ao utilizador correr uma análise completa na sua máquina, ao realizar esta análise é emitido *popups* informando o estado. Por fim é esperado que com um simples click o programa crie automaticamente uma base de dados local, onde será alocado todas as tabelas, como já referido acima, e execute todos os métodos responsáveis por obter os programas instalados ou as bibliotecas *Python* (adição ao projeto) e por fim procede às *queries* à API da NIST, escolhendo os dados recolhidos e alocando-os nas tabelas inicialmente criadas.

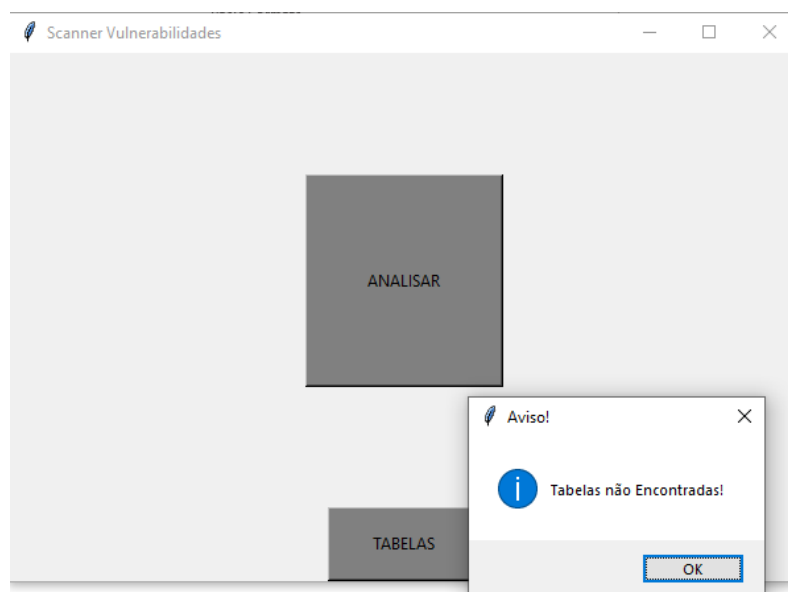


Figura 7 – Interface e *popups* implementados

```
checking cpe:2.3:a:python:certifi:*:*:*:*:*:*
checking cpe:2.3:a:python:chardet:*:*:*:*:*:*
checking cpe:2.3:a:python:idna:*:*:*:*:*:*
checking cpe:2.3:a:python:pip:*:*:*:*:*:*
```

Figura 8 – Processo de Scan, neste caso apenas os módulos de *python*

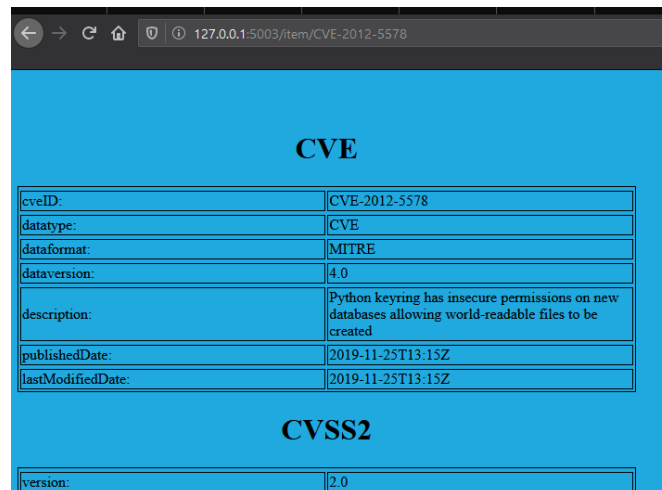
Além disto foi desenvolvido uma outra Interface Gráfica, mas em *flask* com o intuito desta funcionar por uma página Web, foi utilizada a biblioteca *flask-table*, de modo a facilitar o processo de criação de tabelas como também a ligação dos elementos da tabela a outras páginas como links. Como no GUI anterior toda os dados são previamente obtidos através dos métodos desenvolvidos e então retirados da base de dados que é criada localmente.



**Resultados**

CPEID	CVEID
cpe:2.3:a:python:keyring:*.~*~*~*~*~*	<a href="#">CVE-2012-5578</a>
cpe:2.3:a:python:keyring:*.~*~*~*~*~*	<a href="#">CVE-2012-5577</a>
cpe:2.3:a:python:keyring:*.~*~*~*~*~*	<a href="#">CVE-2012-4571</a>
cpe:2.3:a:python:pillow:*.~*~*~*~*~*	<a href="#">CVE-2020-5312</a>
cpe:2.3:a:python:pillow:*.~*~*~*~*~*	<a href="#">CVE-2020-5311</a>
cpe:2.3:a:python:pillow:*.~*~*~*~*~*	<a href="#">CVE-2020-10378</a>
cpe:2.3:a:python:pillow:*.~*~*~*~*~*	<a href="#">CVE-2020-11538</a>
cpe:2.3:a:python:pillow:*.~*~*~*~*~*	<a href="#">CVE-2020-10994</a>
cpe:2.3:a:python:pillow:*.~*~*~*~*~*	<a href="#">CVE-2020-10379</a>
cpe:2.3:a:python:pillow:*.~*~*~*~*~*	<a href="#">CVE-2020-10177</a>
cpe:2.3:a:python:pillow:*.~*~*~*~*~*	<a href="#">CVE-2020-5313</a>
cpe:2.3:a:python:pillow:*.~*~*~*~*~*	<a href="#">CVE-2019-19911</a>

Figura 9 – Lista de CPE e os respetivo CVEID



**CVE**

cveID:	CVE-2012-5578
datatype:	CVE
dataformat:	MITRE
dataversion:	4.0
description:	Python keyring has insecure permissions on new databases allowing world-readable files to be created
publishedDate:	2019-11-25T13:15Z
lastModifiedDate:	2019-11-25T13:15Z

**CVSS2**

version:	2.0
----------	-----

Figura 10 – Informação Detalhada do CVE

## Inviabilidade do método

Como referido em cima, os CPE foram criados para que com a informação fornecida pelo software fosse possível criar um CPE que pudesse depois corresponder a um possível CVE, mas em prática não é tão fácil de implementar.

### Os nomes não correspondem ao que encontramos no CPE

Vamos ver um exemplo de um programa do Windows

Esta é a informação que é possível obter, através de uma procura no registo com as funções criadas:

```
name: Microsoft Visual C++ 2008 Redistributable - x64 9.0.30729.6161
version:9.0.30729.6161
publisher:Microsoft Corporation
```

Figura 1 – Exemplo de um programa detetado pelos nossos métodos

Este é o CPE de referência:

Version 2.3: **cpe:2.3:a:microsoft:visual\_c|+|+2008:sp1:\*:\*:\*:\*:**

Figura 2 – CPE de Referência

Se nós dividirmos o CPE em partes lógicas teríamos:

**Vendor** – *Microsoft*

**Product** – *Visual C++*

**Version** – *2008*

**Update** – *sp1*

Como podemos ver, algumas das informações não são compatíveis, sendo que o nome do vendedor é facilmente arranjado, um dos problemas é a versão.

O NIST assume a versão do programa como sendo 2008, enquanto que o Windows especifica a versão como 9.0.30729.6161

Agora, se este fosse um caso único, era facilmente arranjado para poder corresponder aos parâmetros do NIST, mas o problema é que isto repete-se para múltiplos programas. Por exemplo o WinRAR:

```
name: WinRAR 5.80 (64-bit)
version:5.80.0
publisher:win.rar GmbH
```

Figura 3 – Informações obtidas para o WinRAR

Version 2.3: **cpe:2.3:a:rarlab:winrar:-:\*:\*:\*:\*:\***

Figura 4 – Versão CPE do WinRAR

Podemos observar que o nome do publisher não corresponde com aquele que encontramos no CPE na base de dados da NIST.

Um dos melhores exemplos é o *Launcher* da EPIC GAMES:

```
name: Launcher Prerequisites (x64)
version:1.0.0.0
publisher:Epic Games, Inc.
```

Figura 5 - Informações obtidas para o *Launcher* da EPIC GAMES

Version 2.3: **cpe:2.3:a:epicgames:launcher:8.2.2:\*:\*:\*:\*:\***

Figura 6- Versão CPE do Launcher da EPIC GAMES

Se lermos a especificação de como nomear CPE no documento da NIST [1] podemos observar que todos os espaços devem ser substituídos por um *underscore*, algo que como podemos ver, não acontece, sendo que o espaço é retirado completamente.

Este tipo de nomeação aleatória acontece para muitos CPE, e como tal, torna inviável utilizá-los numa aplicação real. Ainda foi pensado na opção de desenvolver um algoritmo que detetasse tais irregularidades e as corrigisse, usando expressões Regulares ou *Regex*, mas tal não foi possível implementar, visto que este tema daria tema para um outro tema de projeto e nos levaria demasiado tempo a desenvolver.

## 2 Conclusão

Com este trabalho pudemos aplicar diversas matérias lecionadas durante este ano letivo, principalmente Tecnologias de Segurança, abordamos temas como CVE, CPE, Desenvolvimento de Software Seguro entre outros assuntos.

Para concluir, as dificuldades encontradas foram essencialmente pelo fato de ser inviável corresponder um CPE a um CVE, isto referente à nossa abordagem que achamos que seria a melhor opção segundo diversas fontes. Além disso, foi também de difícil compreensão qual o produto que deveríamos desenvolver, devido às diversas possibilidades que este tópico nos “transporta”, e além dos problemas que fomos encontrando e solucionando, alguns ficaram pendentes, porém achamos que de um modo geral conseguimos desenvolver um protótipo funcional, que como o próprio tema sugere – modular – o que nos permitirá ou permitirá a outros a possibilidade de utilizar este protótipo como fonte de estudo ou então melhorar e expandir este projeto que achamos que tem bastante potencial.

## Bibliografia

- [1] NIST, “Common Platform Enumeration: Name Matching Specification Version 2.3”.
- [2] [Online]. Available: [https://csrc.nist.gov/schema/cpe/2.3/cpe-naming\\_2.3.xsd](https://csrc.nist.gov/schema/cpe/2.3/cpe-naming_2.3.xsd).
- [3] B. Byers, “Automation Support for CPE Retrieval,” U.S, 2019.
- [4] “Making-security-measurable,” 27 Maio 2010. [Online]. Available: <http://making-security-measurable.1364806.n2.nabble.com/CPE-automation-and-a-call-for-feedback-td5110199.html>.
- [5] “Making-security-measurable,” Agosto 2013. [Online]. Available: <http://making-security-measurable.1364806.n2.nabble.com/CPE-Common-Platform-Enumeration-f87742.html>.
- [6] “Cpe Mitre,” 22 Março 2013. [Online]. Available: <https://cpe.mitre.org/specification/>.
- [7] T. W. N. Z. Andrew Buttner, “CPE Specification 1.1,” 2007.