VOLTAR

Criação crawlers web e exploits em Python

Apresentar a técnica de crawler, criando sripts para exploração (exploit) como exemplo utilizando funções da biblioteca Python.

NESTE TÓPICO

Marcar tópico

Olá alunos,

A técnica de crawler ou web crawler se assemelha e muitos nem fazem distinção, com a técnica de scraping. O termo de crawler tem como significado rastrear ou rastreador e o termo scrap tem como significado raspar.

Na realidade, o scraping consiste em extrair informações sobre um site. O crawler, também é chamado de spider ou bot (robot) e consiste em identificar as urls, os hiperlinks de um site, para uma futura visita, se necessário.

Como exemplos de crawlers, podemos citar dois tipos de grandes buscadores: o Yahoo! Sluro, o crawler da Yahoo!® e o Googlebot, que é o crawler da Google®.

Lembrando que exploits em certas situações, é um termo utilizado por invasores para explorar dados confidenciais, mas agora vamos utilizar este termo para criar scripts para explorar sites.

Vamos ver um tipo de script, mas antes é necessário utilizar um módulo que não está nativo no Python 3: **lxml**, é uma biblioteca que trabalha com o HTML e XML.

Para instalar, vamos utilizar o **pip install**: no Windows abra o prompt de comando e vá para a pasta **Scripts** que está dentro da pasta **Python**. (utilize o comando **cd nomePasta** para entrar nas pastas (diretórios) e **cd..** para voltar para pasta (diretório) anterior). Utilize também o comando **dir** para verificar quais são os arquivos da pasta corrente.

```
1.   C:\> cd Python36-32
2.  C:\Python36-32> cd Scripts
3.  C:\Python36-32\Scripts>pip install lxml
4.  Collecting lxml
5.     Downloading https://files.pythonhosted.org/packages/47/6d/c350f294c32b152e6a6fb34e
    7aabf2bacfd5f8cc08d59bed0ff3c5dc9cab/lxml-4.2.3-cp36-cp36m-win32.whl (3.2MB)
6.       100% |¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦| 3.2MB 2.9MB/s
7.  Installing collected packages: lxml
8.  Successfully installed lxml-4.2.3
```

Agora vamos ver um exemplo de script:

```
1.  import urllib.request
2.  import bs4 as bs
3.  url = urllib.request.urlopen('http://www.uninove.br').read()
4.  sopa = bs.BeautifulSoup(url, 'lxml')
5.  print(sopa.h1.text)
6.  input('Pressione ENTER para sair...')
```

Na linha **1**, importamos as bibliotecas: **urllib e request** e na linha **2**, importamos o módulo já conhecido BeautifulSoup (**bs4**), a já conhecida 'sopa bonita".

Na linha **3**, passamos para a variável **url** a página da Uninove como exemplo. A função **urlopen** vai abrir a página e pegar todos os dados.

Na linha **5**, após pegarmos os dados da página, utilizamos como filtro, a tag **h1**, que representa o título da página. O resultado foi este:

```
1.  Radar UNINOVE
2.  Pressione ENTER para sair...
```

Vamos modificar um pouco o script:

```
1.  import urllib.request
2.  import bs4 as bs
3.  from pprint import pprint
4.  url = urllib.request.urlopen('http://www.uninove.br').read()
5.  sopa = bs.BeautifulSoup(url, 'lxml')
6.  pprint(sopa.find_all('p'))
7.  input('Pressione ENTER para sair...')
```

Na linha **3**, utilizamos a função **pprint**, que é parecida com a função **print**, só que apresenta um resultado mais formatado. O primeiro **p** de **print** significa a palavra **pretty** (bonita).

Na linha **6**, utilizamos um outro filtro, a tag **p** que pega os parágrafos da página.

E uma parte do resultado foi este:

```
</script>
<script   src="http://www.uninove.br/app/plugins/responsive-lightbox/js/front.js?ver=
1.7.2" type="text/javascript"></script>
<link href="http://www.uninove.br/wp-json/" rel="https://api.w.org/"/>
<link  href="http://www.uninove.br/wp-json/oembed/1.0/embed?url=http%3A%2F%2Fwww.unin
ove.br%2F" rel="alternate" type="application/json+oembed"/>
<link  href="http://www.uninove.br/wp-json/oembed/1.0/embed?url=http%3A%2F%2Fwww.unin
ove.br%2F&amp;format=xml" rel="alternate" type="text/xml+oembed"/>
</p>,
 <p>Informação e conhecimento disponíveis o tempo todo para você!</p>,
  <p><a  href="http://www.uninove.br/biblioteca/sobre-a-biblioteca/apresentacao/">Aces
se.</a></p>,
 <p>Laboratórios das áreas de exatas e biológicas, com recursos modernos que dinamiz
am o aprendizado.</p>,
   <p><a  href="http://www.uninove.br/conheca-a-uninove/estrutura/laboratorios/">Visit
e.</a></p>,
 <p>Primeira Universidade a oferecer wifi ilimitado para mais de 150 mil alunos.</p
>]
Pressione ENTER para sair...
```

Vamos continuar a modificar os filtros:

```
import urllib.request
import bs4 as bs
from pprint import pprint
url = urllib.request.urlopen('http://www.uninove.br').read()
sopa = bs.BeautifulSoup(url, 'lxml')
for cont in sopa.find_all('p'):
    pprint(cont.string)
input('Pressione ENTER para sair...')
```

Na linha **6**, abrimos um loop para percorrer e encontrar todos os parágrafos da página, pois a tag **p** representa todos os parágrafos.

O resultado foi este:

```
'Informação e conhecimento disponíveis o tempo todo para você!'
'Acesse.'
('Laboratórios das áreas de exatas e biológicas, com recursos modernos que '
 'dinamizam o aprendizado.')
'Visite.'
'Primeira Universidade a oferecer wifi ilimitado para mais de 150 mil alunos.'
Pressione ENTER para sair...
```

Vamos continuar a modificar o script:

```
import urllib.request
import bs4 as bs
from pprint import pprint
url = urllib.request.urlopen('http://www.uninove.br').read()
sopa = bs.BeautifulSoup(url, 'lxml')
corpo = sopa.body
for cont in corpo.find_all('p'):
    pprint(cont.string)
input('Pressione ENTER para sair...')
```

Na linha **6**, criamos a variável **corpo** para considerar só a parte principal da página (body) e no caso, os parágrafos da parte principal

E o resultado foi este:

```
'Informação e conhecimento disponíveis o tempo todo para você!'
'Acesse.'
('Laboratórios das áreas de exatas e biológicas, com recursos modernos que '
 'dinamizam o aprendizado.')
'Visite.'
'Primeira Universidade a oferecer wifi ilimitado para mais de 150 mil alunos.'
Pressione ENTER para sair...
```

No caso, o resultado são os mesmos porque todas as strings dos parágrafos estão na parte principal da página.

Vamos continuar utilizando outro filtro:

```
import urllib.request
import bs4 as bs
from pprint import pprint
url = urllib.request.urlopen('http://www.uninove.br').read()
sopa = bs.BeautifulSoup(url, 'lxml')
pprint(sopa.find_all('i'))
input('Pressione ENTER para sair...')
```

Na linha **6**, utilizamos o filtro com a tag **i** que pegou todos os ícones da página e o resultado foi este:

```
[<i class="icon-menu"></i>,
 <i class="icon-search"></i>,
 <i class="icon-person"></i>,
 <i class="icon-radar"></i>,
 <i class="icon-radar"></i>,
 <i class="icon-facebook"></i>,
 <i class="icon-twitter"></i>,
 <i class="icon-google-plus"></i>,
 <i class="icon-facebook"></i>,
 <i class="icon-twitter"></i>,
 <i class="icon-google-plus"></i>,
 <i class="icon-facebook"></i>,
 <i class="icon-twitter"></i>,
 <i class="icon-google-plus"></i>,
 <i class="icon-facebook"></i>,
 <i class="icon-twitter"></i>,
 <i class="icon-google-plus"></i>,
 <i class="icon-facebook"></i>,
 <i class="icon-twitter"></i>,
 <i class="icon-google-plus"></i>,
 <i class="icon-facebook"></i>,
 <i class="icon-twitter"></i>,
 <i class="icon-youtube"></i>,
 <i class="icon-google-plus"></i>,
 <i class="icon2-linkedin"></i>]
Pressione ENTER para sair...
```

Até agora, utilizamos os scripts praticamente para realizar o scraping (raspagem) da página da WEB. Vamos então, tentar utilizar o mesmo tipo de script para fazer um crawling (rastreamento):

```
1.  import urllib.request
2.  import bs4 as bs
3.  from pprint import pprint
4.  url = urllib.request.urlopen('http://www.uninove.br').read()
5.  sopa = bs.BeautifulSoup(url, 'lxml')
6.  for cont in sopa.find_all('a'):
7.      pprint(cont.string)
8.  input('Pressione ENTER para sair...')
```

Na linha **6**, criamos um loop com a tag **a** que vai trazer os links da página para outros recursos da própria página e o resultado foi este:

```
1.  'Processo Seletivo'
2.  'GRADUAÇÃO'
3.  'MESTRADO E DOUTORADO'
4.  'ESPECIALIZAÇÃO, MBA E APRIMORAMENTO'
5.  'EAD'
6.  'MEDICINA'
7.  'Resultados e matrículas'
8.  'RESIDÊNCIAS EM SAÚDE'
9.  'Cursos de Docência'
10. 'ENEM'
11. 'Cursos'
12. 'Graduação'
13. 'EAD'
14. 'Especialização e MBA'
15. 'Cursos de aprimoramento'
16. 'Medicina'
17. 'Mestrado e Doutorado'
18. 'Colégio e Cursos Técnicos'
19. 'Cursos de Docência'
20. 'Cursos Livres'
```

Continuando com o rastreamento da página:

```
1.  import urllib.request
2.  import bs4 as bs
3.  from pprint import pprint
4.  url = urllib.request.urlopen('http://www.uninove.br').read()
5.  sopa = bs.BeautifulSoup(url, 'lxml')
6.  for cont in sopa.find_all('a'):
7.      pprint(cont.get('href'))
8.  input('Pressione ENTER para sair...')
```

Na linha **7**, além de pegar os links da página para outros recursos, também trouxe todos os links do site com a tag **href**.

Vamos ver o resultado:

```
'#panel-main-selective-proccess-mobile'
'https://seletivo.uninove.br/graduacao-e-curta-duracao/todos/todos/#utm_source=portal&utm_medium=menu-seletivo&utm_campaign=padrao&utm_content=link-seletivo'
'http://www.uninove.br/mestrado-e-doutorado/'
'http://www.uninove.br/inscricao-de-pos-graduacao/'
'https://seletivo.uninove.br/graduacao-e-curta-duracao/todos/todos/#utm_source=portal&utm_medium=menu-seletivo-ead&utm_campaign=padrao&utm_content=link-seletivo'
'http://www.uninove.br/inscricoes-de-medicina/'
'http://www.uninove.br/processo-seletivo/resultado-e-matriculas/'
'http://www.uninove.br/residencia-saude/'
'http://www.uninove.br/docencia/'
'http://www.uninove.br/processo-seletivo/usando-a-nota-do-enem/'
'#panel-main-courses-mobile'
'http://www.uninove.br/graduacao/'
'http://www.uninove.br/ead/'
'http://www.uninove.br/pos-graduacao/'
'http://www.uninove.br/aprimoramento/'
'http://www.uninove.br/graduacao/medicina/'
'http://www.uninove.br/mestrado-e-doutorado/'
'http://www.uninove.br/colegio/'
'http://www.uninove.br/docencia/'
'http://www.uninove.br/cursos-livres/'
```

Este tipo de resultado está mais ligado à técnica de crawler, pegamos as urls da página e podemos armazena-las para futuras pesquisas.

Vamos continuar:

```
import urllib.request
import bs4 as bs
from pprint import pprint
url = urllib.request.urlopen('http://www.uninove.br').read()
sopa = bs.BeautifulSoup(url, 'lxml')
navegacao = sopa.nav
for cont in navegacao.find_all('a'):
    print(cont.get('href'))

input('Pressione ENTER para sair...')
```

Na linha **6**, criamos uma variável **navegacao** para receber o parâmetro **nav**, ligado a tag **nav** que está ligada aos links de navegação da página. E com isto, na linha **8**, com o print em loop, imprimimos todos as urls de navegação do site da Uninove.

O resultado foi este:

```
https://seletivo.uninove.br/graduacao-e-curta-duracao/todos/todos/#utm_source=portal&utm_medium=menu-seletivo&utm_campaign=padrao&utm_content=link-seletivo
http://www.uninove.br/mestrado-e-doutorado/
http://www.uninove.br/inscricao-de-pos-graduacao/
https://seletivo.uninove.br/graduacao-e-curta-duracao/todos/todos/#utm_source=portal&utm_medium=menu-seletivo-ead&utm_campaign=padrao&utm_content=link-seletivo
http://www.uninove.br/inscricoes-de-medicina/
http://www.uninove.br/processo-seletivo/resultado-e-matriculas/
http://www.uninove.br/residencia-saude/
http://www.uninove.br/docencia/
http://www.uninove.br/processo-seletivo/usando-a-nota-do-enem/
Pressione ENTER para sair...
```

Atualmente existem diversas ferramentas que fazem este tipo de processamento. Na realidade, isto está ligado a filtros utilizados em scraping ou em crawler, como já comentado a diferença entre estas duas técnicas é

muito tênue, pois o objetivo é colher informações para as pesquisas necessárias.

SAIBA MAIS...

Dê uma olhada nos links abaixo para saber mais sobre a linguagem Python:

https://www.python.org/doc/ (https://www.python.org/doc/)

https://wiki.python.org/moin/PythonBooks (https://wiki.python.org/moin/PythonBooks)

Neste tópico vimos a técnica de crawler e a pequena diferença com a do scraping em páginas da WEB. Aplicamos exemplos com scripts, que através do Pyhton podem ser criados até de forma simples, utilizando a sua biblioteca.

Quiz

Exercício Final

Criação crawlers web e exploits em Python

INICIAR

Referências

SUMMERFIELD, M. *Programação em Python 3*: Uma introdução completa à linguagem Python. Rio de Janeiro Alta Books, 2012. 495 p.

MENEZES, N. N. C. *Introdução à programação com Python:* algoritmos e lógica de programação para iniciantes. 2. ed. São Paulo: Novatec, 2014. 328 p.

SWEIGART, AL. *Automatize tarefas maçantes com Python:* programação prática para verdadeiros iniciantes. São Paulo: Novatec, 2015. 568 p.

PYTHON, doc. Disponível em: <https://www.python.org/doc/>. Acesso em: Junho/2018.

PYTHON, books. Disponível em: <https://wiki.python.org/moin/PythonBooks>. Acesso em: Junho/2018.

Avalie este tópico

ANTERIOR
Auditar aplicações WEB

Índice

Biblioteca (https://www.uninove.br/conheca-a-uninove/biblioteca/sobre-a-biblioteca/apresentacao/)
Portal Uninove (http://www.uninove.br)
Mapa do Site

Ajuda? (https://ava.unidCurso=)