

```
{
  "cells": [
    {
      "cell_type": "code",
      "execution_count": 1,
      "metadata": {},
      "outputs": [
        {
          "name": "stdout",
          "output_type": "stream",
          "text": [
            "Alessandro Parisi \n",
            "last updated: 2019-02-19 \n",
            "\n",
            "CPython 3.5.4\n",
            "IPython 6.1.0\n",
            "\n",
            "numpy 1.16.1\n",
            "pandas 0.20.3\n",
            "matplotlib 2.0.2\n",
            "sklearn 0.20.0\n",
            "seaborn 0.8.0\n"
          ]
        }
      ]
    }
  ]
}
```

```
],  
  
"source": [  
  
    "%load_ext watermark\n",  
  
    "%watermark -a \"Alessandro Parisi\" -u -d -v -p  
numpy,pandas,matplotlib,sklearn,seaborn\n",  
  
    "# to install watermark launch 'pip install watermark' at command line"  
  
]  
  
},  
  
{  
  
    "cell_type": "code",  
  
    "execution_count": 2,  
  
    "metadata": {  
  
        "collapsed": true  
  
    },  
  
    "outputs": [],  
  
    "source": [  
  
        "import pandas as pd\n",  
  
        "import numpy as np\n",  
  
        "from sklearn import *\n",  
  
        "from sklearn.linear_model import LogisticRegression\n",  
  
        "from sklearn.metrics import accuracy_score\n",  
  
        "import warnings \n",  
  
        "warnings.simplefilter('ignore')\n"  
  
    ]
```

```

},

{
  "cell_type": "code",
  "execution_count": 3,
  "metadata": {
    "collapsed": true
  },
  "outputs": [],
  "source": [
    "phishing_dataset = np.genfromtxt('../datasets/phishing_dataset.csv',
delimiter=',', dtype=np.int32)\n",
    "samples = phishing_dataset[:, :-1]\n",
    "targets = phishing_dataset[:, -1]"
  ]
},

{
  "cell_type": "code",
  "execution_count": 4,
  "metadata": {
    "collapsed": true
  },
  "outputs": [],
  "source": [
    "from sklearn.model_selection import train_test_split\n",

```

```

"\n",

"training_samples, testing_samples, training_targets, testing_targets =
train_test_split(\n",

"        samples, targets, test_size=0.2, random_state=0)"

]

},

{

"cell_type": "code",

"execution_count": 5,

"metadata": {

"collapsed": true

},

"outputs": [],

"source": [

"log_classifier = LogisticRegression()"

]

},

{

"cell_type": "code",

"execution_count": 6,

"metadata": {},

"outputs": [

{

"data": {

```

```

"text/plain": [
    "LogisticRegression(C=1.0, class_weight=None, dual=False,
fit_intercept=True,\n",
    "    intercept_scaling=1, max_iter=100, multi_class='warn',\n",
    "    n_jobs=None, penalty='l2', random_state=None,
solver='warn',\n",
    "    tol=0.0001, verbose=0, warm_start=False)"
]
},
"execution_count": 6,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
    "log_classifier.fit(training_samples, training_targets)"
]
},
{
    "cell_type": "code",
    "execution_count": 7,
    "metadata": {
        "collapsed": true
    },
    "outputs": [],

```

```
"source": [  
    "predictions = log_classifier.predict(testing_samples)"  
]  
,  
{  
    "cell_type": "code",  
    "execution_count": 8,  
    "metadata": {},  
    "outputs": [  
        {  
            "name": "stdout",  
            "output_type": "stream",  
            "text": [  
                "Logistic Regression accuracy: 91.72320217096338\n"  
            ]  
        }  
    ],  
    "source": [  
        "accuracy = 100.0 * accuracy_score(testing_targets, predictions)\n",  
        "print (\"Logistic Regression accuracy: \" + str(accuracy))"  
    ]  
},  
{  
    "cell_type": "code",
```

```
"execution_count": null,

"metadata": {

  "collapsed": true

},

"outputs": [],

"source": []

}

],

"metadata": {

  "kernel_spec": {

    "display_name": "Py35",

    "language": "python",

    "name": "py35"

  },

  "language_info": {

    "codemirror_mode": {

      "name": "ipython",

      "version": 3

    },

    "file_extension": ".py",

    "mimetype": "text/x-python",

    "name": "python",

    "nbconvert_exporter": "python",

    "pygments_lexer": "ipython3",
```

```
"version": "3.5.4"

}

},

"nbformat": 4,

"nbformat_minor": 2

}

{

"cells": [

{

"cell_type": "code",

"execution_count": 1,

"metadata": {},

"outputs": [

{

"name": "stdout",

"output_type": "stream",

"text": [

"Alessandro Parisi \n",

"last updated: 2019-02-20 \n",

"\n",

"CPython 3.5.4\n",

"IPython 6.1.0\n",

"\n",

"numpy 1.16.1\n",
```



```
"pandas 0.20.3\n",  
  
"matplotlib 2.0.2\n",  
  
"sklearn 0.20.0\n",  
  
"seaborn 0.8.0\n"  
  
]  
  
}  
  
],  
  
"source": [  
  
"%load_ext watermark\n",  
  
"%watermark -a \"Alessandro Parisi\" -u -d -v -p  
numpy,pandas,matplotlib,sklearn,seaborn\n",  
  
"# to install watermark launch 'pip install watermark' at command line"  
  
]  
  
},  
  
{  
  
"cell_type": "code",  
  
"execution_count": 2,  
  
"metadata": {  
  
"collapsed": true  
  
},  
  
"outputs": [],  
  
"source": [  
  
"import matplotlib.pyplot as plt\n",  
  
"import csv\n",
```

```

"from textblob import TextBlob\n",

"import pandas\n",

"import sklearn\n",

"import numpy as np\n",

"\n",

"import nltk\n",

"# nltk.download('popular')\n",

"# nltk.download('punkt')\n",

"# nltk.download('averaged_perceptron_tagger')\n",

"# nltk.download('wordnet')\n",

"\n",

"from sklearn.feature_extraction.text import CountVectorizer,
TfidfTransformer\n",

"from sklearn.naive_bayes import MultinomialNB\n",

"from sklearn.metrics import classification_report, accuracy_score\n",

"from sklearn.model_selection import train_test_split\n",

"\n",

"from defs import get_tokens\n",

"from defs import get_lemmas"

]

},

{

"cell_type": "code",

"execution_count": 3,

```

```

"metadata": {
    "collapsed": true
},
"outputs": [],
"source": [
    "sms = pandas.read_csv('../datasets/sms_spam_no_header.csv', sep=',',
names=[\"type\", \"text\"])"
    ]
},
{
    "cell_type": "code",
    "execution_count": 4,
    "metadata": {
        "collapsed": true
    },
    "outputs": [],
    "source": [
        "text_train, text_test, type_train, type_test = train_test_split(sms['text'],
sms['type'], test_size=0.3)"
    ]
},
{
    "cell_type": "code",
    "execution_count": 5,
    "metadata": {

```

```
    "collapsed": true
  },
  "outputs": [],
  "source": [
    "bow = CountVectorizer(analyzer=get_lemmas).fit(text_train)"
  ]
},
{
  "cell_type": "code",
  "execution_count": 6,
  "metadata": {
    "collapsed": true
  },
  "outputs": [],
  "source": [
    "sms_bow = bow.transform(text_train)"
  ]
},
{
  "cell_type": "code",
  "execution_count": 7,
  "metadata": {
    "collapsed": true
  },
  "outputs": []
}
```

```
"outputs": [],

"source": [

    "tfidf = TfidfTransformer().fit(sms_bow)"

]

},

{

    "cell_type": "code",

    "execution_count": 8,

    "metadata": {

        "collapsed": true

    },

    "outputs": [],

    "source": [

        "sms_tfidf = tfidf.transform(sms_bow)"

    ]

},

{

    "cell_type": "code",

    "execution_count": 9,

    "metadata": {

        "collapsed": true

    },

    "outputs": [],

    "source": [
```

```

    "spam_detector = MultinomialNB().fit(sms_tfidf, type_train)"
]
},
{
    "cell_type": "code",
    "execution_count": 10,
    "metadata": {},
    "outputs": [
        {
            "name": "stdout",
            "output_type": "stream",
            "text": [
                "predicted: ham\n",
                "expected: ham\n"
            ]
        }
    ],
    "source": [
        "msg = sms['text'][25]\n",
        "msg_bow = bow.transform([msg])\n",
        "msg_tfidf = tfidf.transform(msg_bow)\n",
        "\n",
        "print ('predicted:', spam_detector.predict(msg_tfidf)[0])\n",
        "print ('expected:', sms.type[25])"
    ]
}

```

```
]
},
{
  "cell_type": "code",
  "execution_count": 11,
  "metadata": {
    "collapsed": true
  },
  "outputs": [],
  "source": [
    "predictions = spam_detector.predict(sms_tfidf)"
  ]
},
{
  "cell_type": "code",
  "execution_count": 12,
  "metadata": {},
  "outputs": [
    {
      "name": "stdout",
      "output_type": "stream",
      "text": [
        "accuracy 0.789797487823635\n"
      ]
    }
  ]
}
```

```

    }
],
"source": [
    "print ('accuracy', accuracy_score(sms['type'][:len(predictions)], predictions))"
]
},
{
    "cell_type": "code",
    "execution_count": 13,
    "metadata": {},
    "outputs": [
        {
            "name": "stdout",
            "output_type": "stream",
            "text": [
                "
                precision    recall  f1-score   support\n",
                "\n",
                "
                ham      0.87      0.90      0.88      3382\n",
                "
                spam      0.13      0.10      0.11      519\n",
                "\n",
                "
                micro avg      0.79      0.79      0.79      3901\n",
                "
                macro avg      0.50      0.50      0.50      3901\n",
                "weighted avg      0.77      0.79      0.78      3901\n",
                "\n"
            ]
        }
    ]
}

```



```

    ]
}
],
"source": [
    "print (classification_report(sms['type'][:len(predictions)], predictions))"
]
}
],
"metadata": {
    "kernel_spec": {
        "display_name": "Py35",
        "language": "python",
        "name": "py35"
    },
    "language_info": {
        "codemirror_mode": {
            "name": "ipython",
            "version": 3
        },
        "file_extension": ".py",
        "mimetype": "text/x-python",
        "name": "python",
        "nbconvert_exporter": "python",
        "pygments_lexer": "ipython3",

```

```
"version": "3.5.4"

}

},

"nbformat": 4,

"nbformat_minor": 2

}
```

## **Chapter 1 瞭解對手**

- 1.1 駭客攻擊的原因和手法
- 1.2 遭受駭客攻擊的後果
- 1.3 小心謹慎、未雨綢繆
- 1.4 瞭解防護重點

## **Chapter 2 瀏覽器端的安全性**

- 2.1 瀏覽器的組成
- 2.2 JavaScript 沙盒
- 2.3 磁碟存取權
- 2.4 Cookies
- 2.5 跨站追蹤

## **Chapter 3 加密**

- 3.1 加密原理
- 3.2 加密金鑰
- 3.3 加密傳輸
- 3.4 靜態加密
- 3.5 完整性檢查

## **Chapter 4 Web 伺服器的安全**

- 4.1 檢驗輸入的內容 58
- 4.2 轉義輸出內容
- 4.3 正確處理資源
- 4.4 具象狀態轉換（REST）

- 4.5 縱深防禦
- 4.6 最小權限原則

## **Chapter 5 安全是一種程序**

- 5.1 應用四眼原則
- 5.2 在流程中套用最小權限原則
- 5.3 盡可能採用自動化作業
- 5.4 不重新發明輪子
- 5.5 保留稽核軌跡
- 5.6 撰寫安全的程式碼
- 5.7 借助工具保護自己
- 5.8 坦誠過失

## **Chapter 6 瀏覽器端的漏洞**

- 6.1 跨站腳本
- 6.2 跨站請求偽造
- 6.3 點擊劫持
- 6.4 跨站腳本引入

## **Chapter 7 網路的漏洞**

- 7.1 中間人漏洞
- 7.2 誤導型漏洞
- 7.3 憑證上的漏洞
- 7.4 竊取加解密金鑰

## **Chapter 8 身分驗證機制的漏洞**

- 8.1 暴力攻擊
- 8.2 單一登入
- 8.3 強化身分驗證能力
- 8.4 多因子身分驗證
- 8.5 生物特徵識別
- 8.6 身分憑據的保存方式
- 8.7 使用者枚舉

## **Chapter 9 Session 管理的漏洞**

### 9.1 Session 的運作原理

### 9.2 Session 劫持

### 9.3 Session 竄改

## **Chapter 10 授權機制的漏洞**

### 10.1 為授權建模

### 10.2 設計授權機制

### 10.3 實作存取控制

### 10.4 測試授權機制

### 10.5 常見的授權缺失

## **Chapter 11 資料載荷上的漏洞**

### 11.1 反序列化攻擊

### 11.2 XML 的漏洞

### 11.3 檔案上傳的漏洞

### 11.4 路徑遍歷

### 11.5 批量賦值

## **Chapter 12 注入型漏洞**

### 12.1 遠端程式碼執行

### 12.2 SQL 注入

### 12.3 NoSQL 注入

### 12.4 LDAP 注入

### 12.5 命令注入

### 12.6 CRLF 注入

### 12.7 Regex 注入

## **Chapter 13 第三方程式裡的漏洞**

### 13.1 依賴項

### 13.2 堆疊的更下層

### 13.3 資訊外洩

### 13.4 不安全的組態

## **Chapter 14 不知情的幫凶**

### 14.1 伺服器端請求偽造（SSRF）

### 14.2 電子郵件詐欺

### 14.3 開放式重導向

## **Chapter 15 遭駭時的處置之道**

### 15.1 知道何時被攻擊

### 15.2 阻止進行中的攻擊

### 15.3 釐清來龍去脈

### 15.4 避免重蹈覆轍

### 15.5 向使用者傳達入侵事件的細節

### 15.6 降低未來被入侵的風險